Thomas L. Floyd

Fundamentos de Sistemas Digitales

7ª edición

Fundamentos de Sistemas Digitales, gracias a su inigualable cobertura de aplicaciones y técnicas de diagnóstico, ha enseñado a los estudiantes durante dos décadas a convertirse en técnicos e ingenieros electrónicos de éxito. Ahora, en una nueva edición, éste superventas de Tom Floyd ofrece a los estudiantes de hoy en día un tratamiento completo y práctico de los circuitos digitales, incluyendo una amplia cobertura de los dispositivos lógicos programables (PLD).

Las nuevas características incluyen:

Las notas informáticas relacionan los temas de electrónica digital con aplicaciones informáticas del mundo real.

Los consejos prácticos ayudan a los estudiantes a entender cómo usar los conceptos presentados en el texto.

El CD-ROM contiene conjuntos de problemas que utilizan el software Electronics Workbench (EWB). Nota: el propio software EWB no se incluye en el CD-ROM.

Características:

- Dos capítulos sobre dispositivos PLD (Capítulo 7 y Capítulo 11).
- Secciones de aplicaciones a los sistemas digitales, con "Prácticas de sistemas" que incluyen tareas de análisis, diseño y diagnóstico.
- Un completo glosario al final del libro.
- Formato a todo color.
- Un ejercicio relacionado para cada ejemplo de trabajo.
- Un resumen y una lista de objetivos al principio de cada capítulo.
- Una introducción y una lista de objetivos al principio de cada sección, dentro de un capítulo.
- Cuestiones de repaso al final de cada sección de capítulo.
- Un resumen al final de cada capítulo.
- Un autotest de respuestas múltiples al final de cada capítulo.
- Numerosos enunciados de problemas.

Visite
el nuevo sitio web de Floyd en
http://www.prenhall.com/Floyd
(en inglés).

Pearson Educación

ISBN 84-205-2994-X

Fundamentos de Sistemas Digitales

7ª edición

Thomas L. Floyd

Incluye CD-ROM

Prentice Hall

Escaneado por FerC

CONSULTORES EDITORIALES:

SEBASTIÁN DORMIDO BENCOMO
Departamento de Informática y Automática
UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

LUIS JOYANES AGUILAR
Departamento de Lenguajes, Sistemas Informáticos e Ingeniería del Software
UNIVERSIDAD PONTIFICIA DE SALAMANCA en Madrid

Fundamentos de Sistemas Digitales

7.ª edición

THOMAS L. FLOYD

Traducción
M.º José Gómez Caño
Ingeniero de Telecomunicación

Revisión técnica
Eduardo Barrera López de Turiso
Departamento de sistemas electrónicos y de control
E.U.I.T. de Telecomunicación
Universidad Politécnica de Madrid

PRENTICE HALL
Madrid • México • Santafé de Bogotá • Buenos Aires • Caracas • Lima • Montevideo
San Juan • San José • Santiago • São Paulo • White Plains

Datos de catalogación bibliográfica

THOMAS L. FLOYD
Fundamentos de Sistemas Digitales. 7.ª edición
PEARSON EDUCACIÓN, S.A., Madrid, 2000

ISBN: **84-205-2994-X**
Materia: Informática 681.3

Formato 195 x 250 Páginas: 1256

THOMAS L. FLOYD
Fundamentos de Sistemas Digitales 7.ª edición

Todos los derechos reservados.
No está permitida la reproducción total o parcial de esta obra ni su tratamiento o transmisión por cualquier medio o método sin autorización escrita de la Editorial.

DERECHOS RESERVADOS
© 2000 respecto a la séptima edición en español por:
PEARSON EDUCACIÓN, S.A.
Núñez de Balboa, 120
28006 Madrid

ISBN: 84-205-2994-X
Depósito Legal: M- 27.046-2000
PRENTICE HALL es un sello editorial autorizado de PEARSON EDUCACIÓN, S.A.

Traducido de: Digital Fundamentals, 7.ª edición
© 2000 por Prentice Hall, Inc
ISBN: 0-13-080-850-4

Edición en español:
Editor: Isabel Capella
Asistente editorial: Ana Isabel García

Cubierta: DIGRAF, S.A.
Composición: REPROGRÁFICAS MALPE, S.A.
Impreso por: Gráficas Rógar, S. A.

IMPRESO EN ESPAÑA - PRINTED IN SPAIN

Este libro ha sido impreso con papel y tintas ecológicos

ÍNDICE DE CONTENIDO

1 INTRODUCCIÓN A LOS CONCEPTOS DIGITALES 1

1.1 Magnitudes analógicas y digitales 4
1.2 Dígitos binarios, niveles lógicos y formas de onda digitales 7
1.3 Operaciones lógicas básicas 15
1.4 Funciones lógicas básicas 17
1.5 Circuitos digitales integrados 24
1.6 Instrumentación para pruebas y localización de averías 28
1.7 Aplicación a los sistemas digitales 38

2 SISTEMAS DE NUMERACIÓN, OPERACIONES Y CÓDIGOS 49

2.1 Números decimales 52
2.2 Números binarios 54
2.3 Conversión decimal-binario 58
2.4 Aritmética binaria 61
2.5 Complemento a 1 y complemento a 2 de los números binarios 65
2.6 Números con signo 67
2.7 Operaciones aritméticas de números con signo 74
2.8 Números hexadecimales 82
2.9 Números octales 89
2.10 Código decimal binario (BCD) 92
2.11 Códigos digitales y paridad 95
2.12 Aplicación a los sistemas digitales 104

3 PUERTAS LÓGICAS 121

3.1 El inversor 124
3.2 La puerta AND 127
3.3 La puerta OR 135
3.4 Puerta NAND 141
3.5 La puerta NOR 147
3.6 Puertas OR-exclusiva y NOR-exclusiva 153
3.7 Puertas lógicas integradas 157
3.8 Localización de averías 170
3.9 Aplicación a los sistemas digitales 178

4 ÁLGEBRA DE BOOLE Y SIMPLIFICACIÓN LÓGICA 201

4.1 Operaciones y expresiones booleanas 204
4.2 Leyes y reglas del álgebra de Boole 206
4.3 Teoremas de DeMorgan 211
4.4 Análisis booleano de los circuitos lógicos 216
4.5 Simplificación mediante el álgebra de Boole 218
4.6 Formas estándar de las expresiones booleanas 221
4.7 Expresiones Booleanas y tablas de verdad 229
4.8 Mapas de Karnaugh 233
4.9 Minimización de una suma de productos mediante el mapa de Karnaugh 235
4.10 Minimización de un producto de sumas mediante el mapa de Karnaugh 246
4.11 Mapa de Karnaugh de cinco variables 251
4.12 Aplicación a los sistemas digitales 253

5 LÓGICA COMBINACIONAL 273

5.1 Circuitos lógicos combinacionales especiales 276
5.2 Implementación de la lógica combinacional 282
5.3 La propiedad universal de las puertas NAND y NOR 288
5.4 Lógica combinacional utilizando las puertas NAND y NOR 291
5.5 Funcionamiento con trenes de impulsos 297
5.6 Localización de averías 301
5.7 Aplicación a los sistemas digitales 308

6 FUNCIONES DE LA LÓGICA COMBINACIONAL 329

6.1 Sumadores básicos 332
6.2 Sumadores binarios en paralelo 336
6.3 Comparadores 345
6.4 Decodificadores 350
6.5 Codificadores 360
6.6 Convertidores de código 365
6.7 Multiplexores (selectores de datos) 370
6.8 Demultiplexores 379
6.9 Generadores/comprobadores de paridad 381
6.10 Localización de averías 385
6.11 Aplicación a los sistemas digitales 388

7 INTRODUCCIÓN A LOS DISPOSITIVOS LÓGICOS PROGRAMABLES 415

7.1 Dispositivos lógicos programables (PLD). Clasificación 418
7.2 Matriz lógica programable (PAL, Programmable Array Logic) 421
7.3 Matriz lógica genérica (GAL, Generic Array Logic) 428
7.4 La GAL22V10 432
7.5 La GAL16V8 441
7.6 Programación de los dispositivos lógicos programables (PLD) 445
7.7 Software de los dispositivos PLD 449
7.8 Aplicación a los sistemas digitales 460

8 FLIP-FLOPS Y DISPOSITIVOS RELACIONADOS 475

8.1 Latches 478
8.2 Flip-flops disparados por flanco 485
8.3 Flip-flops maestro-esclavo 499
8.4 Características de operación de los flip-flops 502
8.5 Aplicaciones de los flip-flops 506
8.6 Monoestables 511
8.7 El temporizador 555 517
8.8 Localización de averías 524
8.9 Aplicación a los sistemas digitales 526

9 CONTADORES 547

9.1 Funcionamiento del contador asíncrono 550
9.2 Funcionamiento del contador síncrono 559
9.3 Contador síncrono ascendente/descendente 568
9.4 Diseño de contadores síncronos 573
9.5 Contadores en cascada 584
9.6 Decodificación de contadores 589
9.7 Aplicaciones de los contadores 593
9.8 Localización de averías 598
9.9 Símbolos lógicos con notación de dependencia 604
9.10 Aplicación a los sistemas digitales 606

10 REGISTROS DE DESPLAZAMIENTO 627

10.1 Funciones básicas de los registros de desplazamiento 630
10.2 Registros de desplazamiento con entrada/salida serie 631
10.3 Registros de desplazamiento con entrada serie/salida paralelo 636
10.4 Registros de desplazamiento con entrada paralelo/salida serie 639

10.5 Registros de desplazamiento con entrada/salida paralelo 642
10.6 Registros de desplazamiento bidireccionales 644
10.7 Contadores basados en registro de desplazamiento 647
10.8 Aplicaciones de los registros de desplazamiento 651
10.9 Localización de averías 659
10.10 Símbolos lógicos con notación de dependencia 662
10.11 Aplicación a los sistemas digitales 664

11 APLICACIONES DE LÓGICA SECUENCIAL CON DISPOSITIVOS LÓGICOS PROGRAMABLES 683

11.1 La celda OLMC al completo 686
11.2 Selección del modo de las OLMC 689
11.3 Implementación de registros de desplazamiento con PLD 694
11.4 Implementación de contadores con PLD 699
11.5 Implementación de sistemas mediante PLD 708
11.6 Aplicación a los sistemas digitales 719

12 MEMORIAS 733

12.1 Principios de las memorias semiconductoras 736
12.2 Memorias de acceso aleatorio (RAM) 740
12.3 Memorias de sólo lectura (ROM) 757
12.4 Memorias ROM programables (PROM y EPROM) 764
12.5 Memorias flash 768
12.6 Expansión de memorias 774
12.7 Tipos especiales de memorias 781
12.8 Memorias ópticas y magnéticas 787
12.9 Pruebas y localización de averías 796
12.10 Aplicación a los sistemas digitales 801

13 INTERFACES 823

13.1 Interfaces entre el mundo digital y el analógico 826
13.2 Conversión digital-analógica (D/A) 830
13.3 Conversión analógico-digital (A/D) 838
13.4 Localización de averías en los convertidores DAC y ADC 849
13.5 Interfaces internas de los sistemas 854
13.6 Buses estándar 859
13.7 Aplicación a los sistemas digitales 872

14 INTRODUCCIÓN A LOS MICROPROCESADORES Y COMPUTADORAS 891

14.1 El microprocesador y la computadora 894
14.2 Familias de microprocesadores 900
14.3 El microprocesador y el modelo software 8086/8088 para el procesador Pentium 906
14.4 Programación de microprocesadores 914
14.5 La unidad central de proceso (CPU) 923
14.6 La memoria 930
14.7 Los puertos de entrada/salida (E/S) 934
14.8 Interrupciones 937
14.9 Acceso directo a memoria (DMA) 941

15 TECNOLOGÍAS DE LOS CIRCUITOS INTEGRADOS 953

15.1 Parámetros y características de operación básicas 956
15.2 Circuitos CMOS 966
15.3 Circuitos TTL 973
15.4 Consideraciones prácticas sobre el uso de TTL 978
15.5 Comparación de las prestaciones CMOS y TTL 987
15.6 Circuitos ECL 988
15.7 PMOS, NMOS y E²CMOS 990

Apéndice A
Hojas de características (data sheets) 1005

Apéndice B
Conversiones 1053

Apéndice C 1057

Respuestas a los problemas impares 1063

Glosario 1107

Índice 1121

PRÓLOGO

Esta séptima edición de *Fundamentos de Sistemas Digitales* ofrece una exposición exhaustiva en un formato claro, conciso y cuidadosamente ilustrado. Como las ediciones anteriores, ésta presenta un buen equilibrio de conceptos básicos, tecnología actualizada, aplicaciones prácticas y técnicas de localización de averías. En esta edición, algunos temas se han ampliado o reforzado, y se han añadido numerosas mejoras a lo largo del libro. También se han añadido algunas características completamente nuevas, incluyendo ejercicios para el banco de trabajo electrónico (EWB), notas informáticas y consejos prácticos. Esta edición se ha revisado y comprobado minuciosamente con el fin de asegurar un tratamiento preciso y actualizado.

Probablemente, encontrará más temas que los que se pueden cubrir en un trimestre. Este amplio rango de temas proporciona flexibilidad para diseñar una amplia variedad de cursos sobre tecnología. Por ejemplo, algunos de los temas orientados al diseño o de carácter matemático, o los que tratan de las aplicaciones a los sistemas digitales pueden no ser apropiados para determinados cursos. Otros programas pueden no abordar los dispositivos PLD o no necesitar una introducción a los microprocesadores. Muchos cursos se ocupan de los detalles internos de los circuitos integrados, mientras que otros no cubren en absoluto los temas de nivel de componente. Dichas materias pueden fácilmente omitirse o revisarse sin excesivo detalle, sin que el resto de los temas se vean afectados. No es imprescindible tener conocimientos previos sobre circuitos de transistores para abordar el estudio de este texto.

Nuevo en esta edición

Además de actualizaciones y mejoras generales en el texto y en los gráficos a lo largo de todo el libro, la séptima edición incluye las siguientes características nuevas.

Tutoriales de banco de trabajo electrónico (EWB) En el sitio web hay disponibles tutoriales, en inglés, asociados con muchos de los capítulos del texto, que pueden descargarse y ser empleados por los estudiantes. Estos tutoriales introducen aspectos y elementos del EWB según las necesidades de cada capítulo, y también contienen ejemplos y ejercicios. Los tutoriales se encuentran en la dirección *http://www.prenhall.com/floyd*.

Simulación EWB para las aplicaciones a los sistemas digitales Para muchas de las aplicaciones a sistemas contenidas en el texto, el circuito lógico está simulado en EWB y está disponible en el CD-ROM que acompaña a este libro. Para poder observar el circuito en funcionamiento para una aplicación concreta, basta con que el

lector abra el archivo especificado, conecte los instrumentos del banco de trabajo electrónico necesarios y ejecute la simulación. El CD-ROM contiene software gratuito de demostración de EWB.

Problemas de localización de averías en el banco de trabajo electrónico. Al final de la mayoría de los capítulos, se incluyen nuevos problemas de localización de averías que hacen referencia a los circuitos contenidos en el CD-ROM. Generalmente, estos circuitos tienen fallos ocultos y los estudiantes deben aplicar técnicas de localización de averías para identificar dichos fallos. En algunos casos, los circuitos no tienen fallos y funcionan correctamente, lo que también deberá determinar el estudiante.

Notas informáticas. Estas notas proporcionan información interesante e instructiva sobre tecnología informática en relación con el tema que se esté tratando. Estas notas informáticas opcionales están identificadas mediante un logotipo y un diseño especiales, y están distribuidas a lo largo del libro, conteniendo la mayoría de los capítulos una o más.

Consejos prácticos. Esta característica proporciona información útil y práctica, intercalada a lo largo del libro. Los consejos prácticos se identifican mediante un logotipo y un diseño especiales. Generalmente, están relacionados con el texto, aunque pueden saltarse sin que ello afecte a las exposiciones del profesor o a la comprensión del tema por parte de los estudiantes.

Otras características nuevas o mejoradas son:

- Una lista de palabras clave al final de cada capítulo.
- Revisiones y mejoras en el contenido del libro.
- Tutoriales revisados sobre el uso de la calculadora, empleando la calculadora TI-85.
- Una exposición revisada de las puertas lógicas integradas en el Capítulo3.
- Una exposición revisada sobre las memorias y dispositivos de almacenamiento en el Capítulo 12.
- Una exposición revisada sobre los buses estándar en el Capítulo 13.
- Una exposición revisada sobre microprocesadores y computadoras en el Capítulo 14.

Características adicionales

- Dos nuevos capítulos sobre dispositivos PLD (Capítulo 7 y Capítulo 11).
- Un capítulo "flotante", el Capítulo 15, que proporciona información sobre la tecnología de circuitos integrados, y que se puede estudiar en cualquier momento que se desee.
- Al comienzo de cada capítulo se enumeran los objetivos del mismo y se hace una breve introducción.
- En cada sección se hace una introducción y una enumeración de los objetivos de la misma.
- Al final de cada sección se plantean una serie de cuestiones.
- Se incluye un problema relacionado en cada ejemplo.
- Hay secciones dedicadas a las aplicaciones de los sistemas digitales al final de la mayor parte de los capítulos.

- Se incluye un resumen al final de cada capítulo.
- Se incluye un autotest al final de cada capítulo.
- Se plantean una gran cantidad de problemas clasificados por secciones.
- Un amplio glosario al final del libro.

Descripción y estructura de los capítulos

Introducción del capítulo. Las primeras páginas de cada capítulo tienen el formato que se indica en la Figura P.1. La página de la izquierda contiene la lista de los objetivos del capítulo, una introducción, y una lista de los dispositivos específicos que se verán en el capítulo (cada nuevo dispositivo se indica mediante el logotipo de un circuito integrado en el lugar donde se introduce). En la página de la derecha se presenta una breve descripción de la aplicación a los sistemas digitales que se verá en el capítulo.

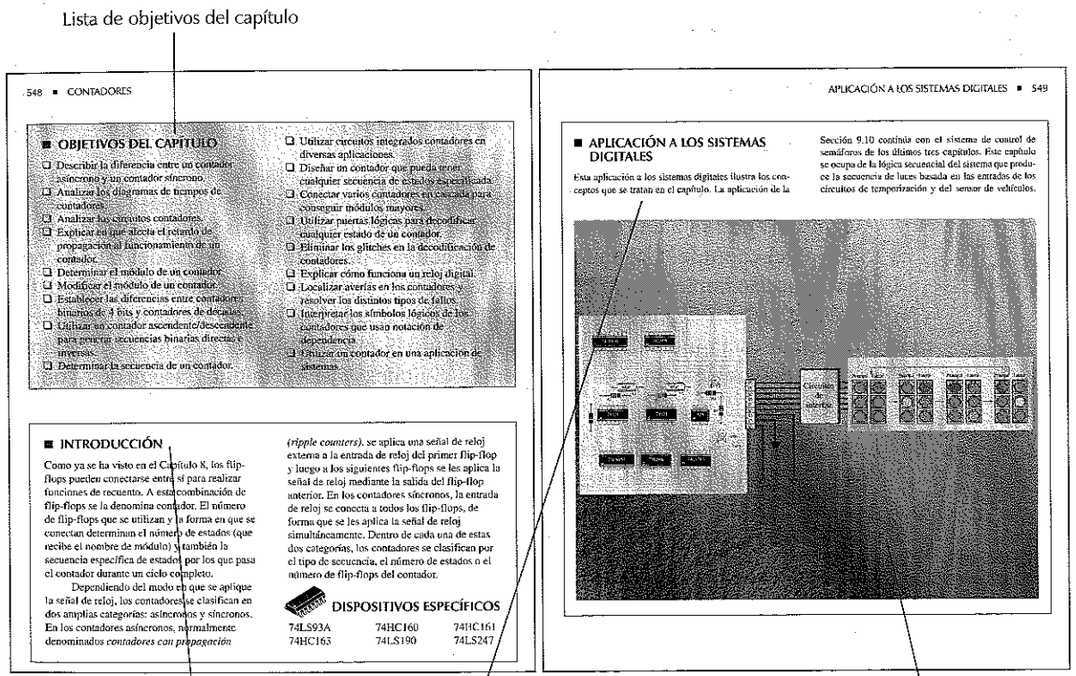


FIGURA P.1
Introducción del capítulo.

Introducción de la sección. Cada sección del capítulo comienza con una breve introducción, que proporciona una visión general y una lista de los objetivos de la misma. En la Figura P.2 se muestra un ejemplo.

Revisión de la sección. Cada sección termina con una revisión, en la que se incluyen preguntas o ejercicios sobre los principales conceptos presentados, como se muestra en la Figura P.2. Las respuestas a estos ejercicios se encuentran al final de cada capítulo.

Ejemplo resueltos y problemas relacionados. Numerosos ejemplos resueltos ayudan a ilustrar y clarificar los conceptos básicos o procedimientos específicos. Cada ejemplo concluye con un problema relacionado que le refuerza o amplia. Algunos problemas relacionados son una repetición del ejemplo, pero utilizando parámetros o condiciones diferentes. Otros están enfocados sobre una parte limitada del ejemplo, para reforzar el concepto. En la Figura P.3 se muestra una página con un ejemplo resuelto típico y un problema relacionado. Las respuestas de todos los problemas relacionados se encuentran al final del libro

Sección de localización de averías. Muchos capítulos incluyen una sección dedicada a la localización de averías, que hace referencia a los temas cubiertos en el capítulo y que se centra en las técnicas de localización de averías y el uso de instrumentos de prueba. En la Figura P.4 se muestra una parte de una sección típica sobre la localización de averías.

Aplicación a los sistemas digitales. La última sección de la mayor parte de los capítulos presenta una aplicación práctica sobre los conceptos y dispositivos cubiertos en el capítulo. Cada una de estas secciones presenta un sistema del mundo real, en el que se implementan las etapas de análisis, diseño y localización de averías a través de las actividades de las "Prácticas de sistemas". Algunas de las aplicaciones a sistemas están limitadas a un único capítulo, y otras se extienden a lo largo de dos o más. Las aplicaciones a los sistemas digitales y sus capítulos asociados son las siguientes:

- Sistema de control y recuento de pastillas: Capítulos 1, 2, 3 y 4.
- Sistema de control digital de un motor: Capítulo 5.
- Sistema de control de semáforos: Capítulos 6, 7, 8, 9 y 11.
- Sistema de seguridad de acceso: Capítulos 10 y 12.
- Sistema de posicionamiento de antena de satélite: Capítulo 13.

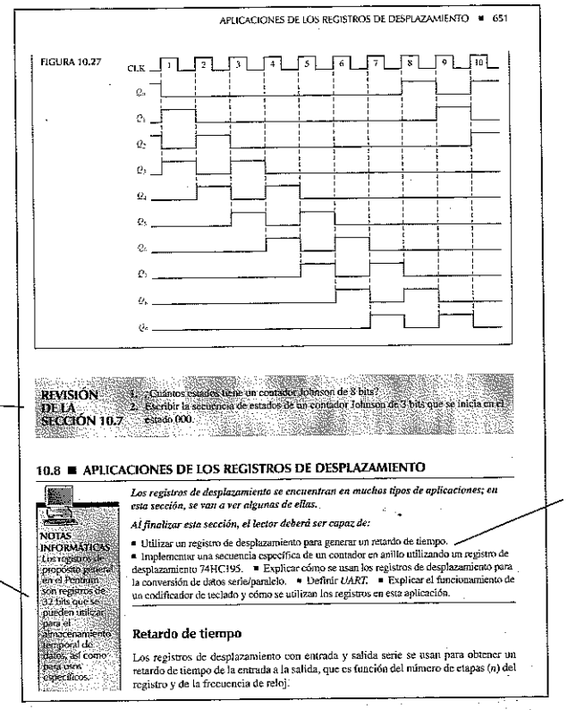

APLICACIONES DE LOS REGISTROS DE DESPLAZAMIENTO ■ 651

FIGURA 10.27

REVISIÓN DE LA SECCIÓN 10.7
1. ¿Cuántos estados tiene un contador Johnson de 8 bits?
2. Escribir la secuencia de estados de un contador Johnson de 3 bits que se inicia en el estado 000.

10.8 ■ APLICACIONES DE LOS REGISTROS DE DESPLAZAMIENTO

Los registros de desplazamiento se encuentran en muchos tipos de aplicaciones; en esta sección, se van a ver algunas de ellas.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Utilizar un registro de desplazamiento para generar un retardo de tiempo. ■ Implementar una secuencia específica de un contador en anillo utilizando un registro de desplazamiento 74HC195. ■ Explicar cómo se usan los registros de desplazamiento para la conversión de datos serie/paralelo. ■ Definir *UART*. ■ Explicar el funcionamiento de un codificador de teclado y cómo se utilizan los registros en esta aplicación.

NOTAS INFORMÁTICAS
Los registros de propósito general en el Pentium son registros de 32 bits que se pueden utilizar para el almacenamiento temporal de datos, así como para usos específicos.

Retardo de tiempo

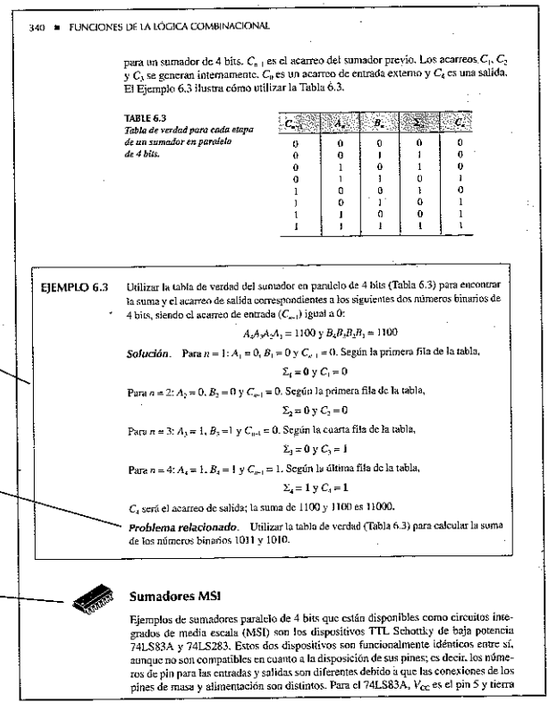
Los registros de desplazamiento con entrada y salida serie se usan para obtener un retardo de tiempo de la entrada a la salida, que es función del número de etapas (n) del registro y de la frecuencia de reloj.

Ejercicios de revisión al final de cada sección.

Las notas informáticas se encuentran a lo largo del texto (no son parte de la sección de introducción).

Párrafo de introducción y lista de los objetivos de la sección al comienzo de la misma.

FIGURA P.2
Introducción y revisión de una sección.


340 ■ FUNCIONES DE LA LÓGICA COMBINACIONAL

para un sumador de 4 bits. C_{n-1} es el acarreo del sumador previo. Los acarreos C_1, C_2 y C_3 se generan internamente. C_4 es un acarreo de entrada externo y C_4 es una salida. El Ejemplo 6.3 ilustra cómo utilizar la Tabla 6.3.

TABLE 6.3
Tabla de verdad para cada etapa de un sumador en paralelo de 4 bits.

C_{n-1}	A_n	B_n	Σ_n	C_n
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

EJEMPLO 6.3 Utilizar la tabla de verdad del sumador en paralelo de 4 bits (Tabla 6.3) para encontrar la suma y el acarreo de salida correspondientes a los siguientes dos números binarios de 4 bits, siendo el acarreo de entrada (C_{n-1}) igual a 0:

$$A_4A_3A_2A_1 = 1100 \text{ y } B_4B_3B_2B_1 = 1100$$

Solución. Para $n = 1$: $A_1 = 0$, $B_1 = 0$ y $C_{n-1} = 0$. Según la primera fila de la tabla,

$$\Sigma_1 = 0 \text{ y } C_1 = 0$$

Para $n = 2$: $A_2 = 0$, $B_2 = 0$ y $C_{n-1} = 0$. Según la primera fila de la tabla,

$$\Sigma_2 = 0 \text{ y } C_2 = 0$$

Para $n = 3$: $A_3 = 1$, $B_3 = 1$ y $C_{n-1} = 0$. Según la cuarta fila de la tabla,

$$\Sigma_3 = 0 \text{ y } C_3 = 1$$

Para $n = 4$: $A_4 = 1$, $B_4 = 1$ y $C_{n-1} = 1$. Según la última fila de la tabla,

$$\Sigma_4 = 1 \text{ y } C_4 = 1$$

C_4 será el acarreo de salida; la suma de 1100 y 1100 es 11000.

Problema relacionado. Utilizar la tabla de verdad (Tabla 6.3) para calcular la suma de los números binarios 1011 y 1010.

Sumadores MSI

Ejemplos de sumadores paralelo de 4 bits que están disponibles como circuitos integrados de media escala (MSI) son los dispositivos TTL Schottky de baja potencia 74LS83A y 74LS283. Estos dos dispositivos son funcionalmente idénticos entre sí, aunque no son compatibles en cuanto a la disposición de sus pines; es decir, los números de pin para las entradas y salidas son diferentes debido a que las conexiones de los pines de masa y alimentación son distintos. Para el 74LS83A, V_{CC} es el pin 5 y tierra

Cada ejemplo queda delimitado mediante un recuadro.

Cada ejemplo contiene un problema relacionado con el mismo.

El icono del circuito integrado indica la introducción de nuevos dispositivos.

FIGURA P.3
Un ejemplo y un problema relacionado.

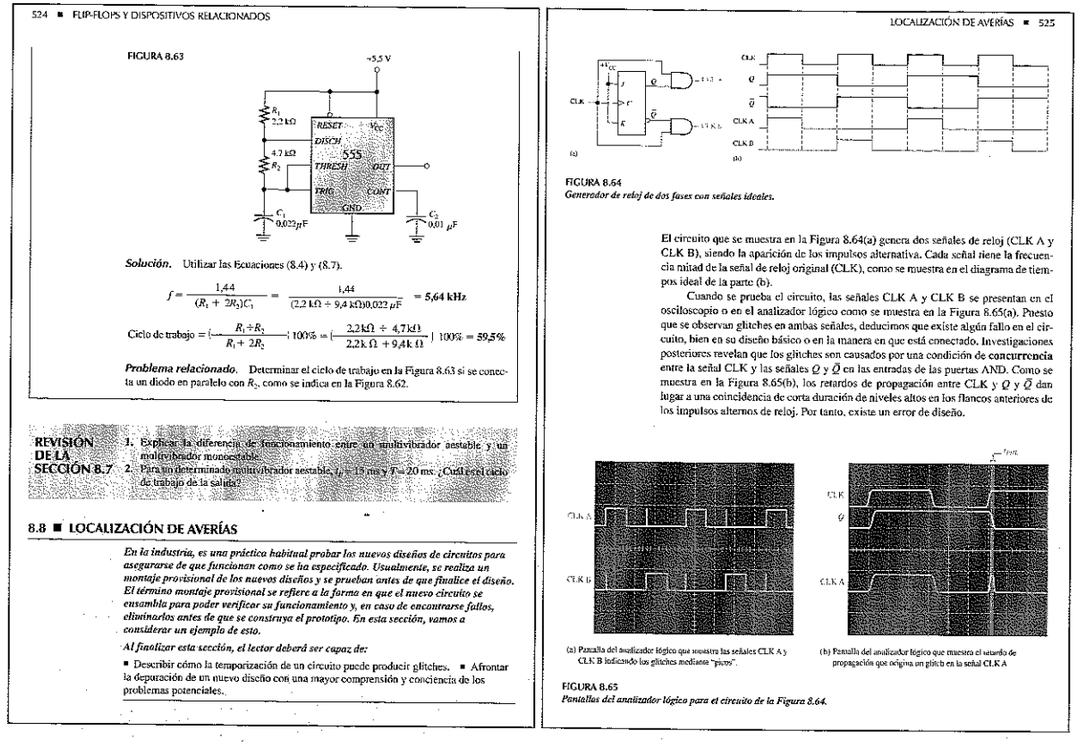

524 ■ FLIP-FLOPS Y DISPOSITIVOS RELACIONADOS

FIGURA 8.63

+5.5 V
R_1 2.2 kΩ
4.7 kΩ
R_2
RESET
DISCH
555
THRESH
OUT
TRIG
CONT
GND
V_{CC}
C_1 0.022 μF
C_2 0.01 μF

Solución. Utilizar las ecuaciones (8.4) y (8.7).

$$f = \frac{1{,}44}{(R_1 + 2R_2)C_1} = \frac{1{,}44}{(2{,}2\,\text{k}\Omega + 9{,}4\,\text{k}\Omega)0{,}022\,\mu\text{F}} = \mathbf{5{,}64\ kHz}$$

$$\text{Ciclo de trabajo} = \left(\frac{R_1 + R_2}{R_1 + 2R_2}\right) 100\% = \left(\frac{2{,}2\,\text{k}\Omega + 4{,}7\,\text{k}\Omega}{2{,}2\,\text{k}\,\Omega + 9{,}4\,\text{k}\,\Omega}\right) 100\% = \mathbf{59{,}5\%}$$

Problema relacionado. Determinar el ciclo de trabajo en la Figura 8.63 si se conecta un diodo en paralelo con R_2, como se indica en la Figura 8.62.

REVISIÓN DE LA SECCIÓN 8.7

1. Explicar la diferencia de funcionamiento entre un multivibrador astable y un multivibrador monoestable.
2. Para un determinado multivibrador astable, $t_1 = 15$ ms y $T = 20$ ms. ¿Cuál es el ciclo de trabajo de la salida?

8.8 ■ LOCALIZACIÓN DE AVERÍAS

En la industria, es una práctica habitual probar los nuevos diseños de circuitos para asegurarse de que funcionan como se ha especificado. Usualmente, se realiza un montaje provisional de los nuevos diseños y se prueban antes de que finalice el diseño. El término montaje provisional se refiere a la forma en que el nuevo circuito se ensambla para poder verificar su funcionamiento y, en caso de encontrarse fallos, eliminarlos antes de que se construya el prototipo. En esta sección, vamos a considerar un ejemplo de esto.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir cómo la temporización de un circuito puede producir glitches. ■ Afrontar la depuración de un nuevo diseño con una mayor comprensión y conciencia de los problemas potenciales.

LOCALIZACIÓN DE AVERÍAS ■ 525

FIGURA 8.64
Generador de reloj de dos fases con señales ideales.

El circuito que se muestra en la Figura 8.64(a) genera dos señales de reloj (CLK A y CLK B), siendo la aparición de los impulsos alternativa. Cada señal tiene la frecuencia mitad de la señal de reloj original (CLK), como se muestra en el diagrama de tiempos ideal de la parte (b).

Cuando se prueba el circuito, las señales CLK A y CLK B se presentan en el osciloscopio o en el analizador lógico como se muestra en la Figura 8.65(a). Puesto que se observan glitches en ambas señales, deducimos que existe algún fallo en el circuito, bien en su diseño básico o en la manera en que está conectado. Investigaciones posteriores revelan que los glitches son causados por una condición de **concurrencia** entre la señal CLK y las señales Q y $\bar{Q}$ en las entradas de las puertas AND. Como se muestra en la Figura 8.65(b), los retardos de propagación entre CLK y Q y $\bar{Q}$ dan lugar a una coincidencia de corta duración de niveles altos en los flancos anteriores de los impulsos alternos de reloj. Por tanto, existe un error de diseño.

(a) Pantalla del analizador lógico que muestra las señales CLK A y CLK B indicando los glitches mediante "picos".

(b) Pantalla del analizador lógico que muestra el retardo de propagación que origina un glitch en la señal CLK A

FIGURA 8.65
Pantallas del analizador lógico para el circuito de la Figura 8.64.

FIGURA P.4
Páginas representativas de una sección típica dedicada a la localización de averías.

Aunque no pretenden ser proyectos de laboratorio (excepto el laboratorio de la mente), muchas de las aplicaciones a los sistemas incluyen representaciones reales de las tarjetas de circuito impreso e instrumentos, para proporcionar experiencia en la tarea de relacionar los esquemas con los circuitos reales, identificar encapsulados de circuitos integrados y obtener datos de las lecturas de ciertos instrumentos. Dado que la omisión de estas secciones no afectará al resto de los temas del libro, pueden verse de forma opcional. En la Figura P.5 se muestra una parte de una sección de "Aplicación a los sistemas digitales".

Fin del capítulo. Al final de cada capítulo se incluye la siguiente información:

- Resumen
- Palabras clave
- Autotest
- Un conjunto de problemas, que incluye algunas o todas las categorías siguientes: problemas básicos, problemas sobre localización de averías, problemas sobre aplicaciones a sistemas, problemas de diseño y problemas de localización de averías con el software EWB.
- Respuestas a las revisiones de las secciones.

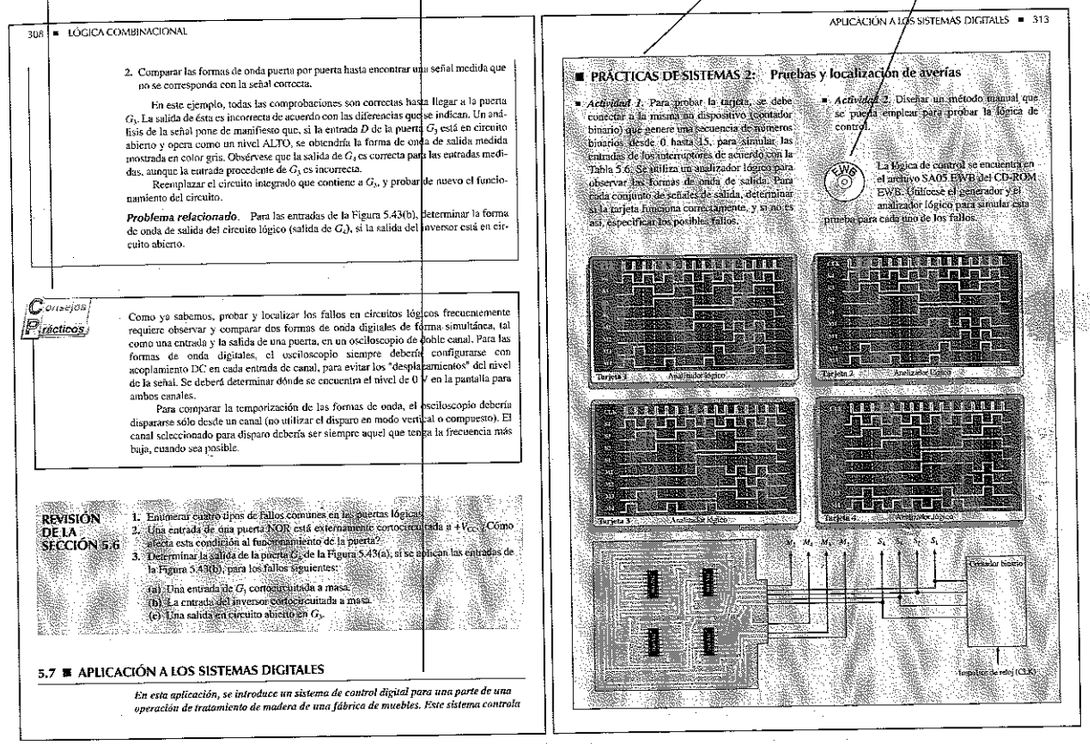
- Respuestas a los problemas relacionados de los ejemplos.
- Respuestas al autotest.

Los consejos prácticos se encuentran a lo largo del texto (no siempre como parte de la sección Aplicación a sistemas digitales).

Introducción a una lista de objetivos.

La sección Práctica de sistemas proporciona una serie de actividades de análisis, diseño o localización de averías. Muchas aplicaciones incluyen una tarjeta de circuito impreso e instrumentación.

Referencia a la simulación EWB contenida en el CD-ROM.


306 ■ LÓGICA COMBINACIONAL

2. Comparar las formas de onda puerta por puerta hasta encontrar una señal medida que no se corresponda con la señal correcta.

En este ejemplo, todas las comprobaciones son correctas hasta llegar a la puerta G_3. La salida de ésta es incorrecta de acuerdo con las diferencias que se indican. Un análisis de la señal pone de manifiesto que, si la entrada D de la puerta G_3 está en circuito abierto y opera como un nivel ALTO, se obtendría la forma de onda de salida medida mostrada en color gris. Obsérvese que la salida de G_3 es correcta para las entradas medidas, aunque la entrada procedente de G_2 es incorrecta.

Reemplazar el circuito integrado que contiene a G_3, y probar de nuevo el funcionamiento del circuito.

Problema relacionado. Para las entradas de la Figura 5.43(b), determinar la forma de onda de salida del circuito lógico (salida de G_4), si la salida del inversor está en circuito abierto.

CONSEJOS Prácticos

Como ya sabemos, probar y localizar los fallos en circuitos lógicos frecuentemente requiere observar y comparar dos formas de onda digitales de forma simultánea, tal como una entrada y la salida de una puerta, en un osciloscopio de doble canal. Para las formas de onda digitales, el osciloscopio siempre deberá configurarse con acoplamiento DC en cada entrada de canal, para evitar los "desplazamientos" del nivel de la señal. Se deberá determinar dónde se encuentra el nivel de 0 V en la pantalla para ambos canales.

Para comparar la temporización de las formas de onda, el osciloscopio debería dispararse sólo desde un canal (no utilizar el disparo en modo vertical o compuesto). El canal seleccionado para disparo debería ser siempre aquel que tenga la frecuencia más baja, cuando sea posible.

REVISIÓN DE LA SECCIÓN 5.6

1. Enumerar cuatro tipos de fallos comunes en las puertas lógicas.
2. Una entrada de una puerta NOR está externamente cortocircuitada a $+V_{CC}$. Cómo afecta esta condición al funcionamiento de la puerta?
3. Determinar la salida de la puerta G_4 de la Figura 5.43(a), si se aplican las entradas de la Figura 5.43(b), para los fallos siguientes:
   (a) Una entrada de G_1 cortocircuitada a masa.
   (b) La entrada del inversor cortocircuitada a masa.
   (c) Una salida en circuito abierto en G_2.

5.7 ■ APLICACIÓN A LOS SISTEMAS DIGITALES

En esta aplicación, se introduce un sistema de control digital para una parte de una operación de tratamiento de madera de una fábrica de muebles. Este sistema controla

APLICACIÓN A LOS SISTEMAS DIGITALES ■ 313

■ **PRÁCTICAS DE SISTEMAS 2: Pruebas y localización de averías**

■ *Actividad 1.* Para probar la tarjeta, se debe conectar a la misma un dispositivo (contador binario) que genere una secuencia de números binarios desde 0 hasta 15, para simular las entradas de los interruptores de acuerdo con la Tabla 5.6. Se utiliza un analizador lógico para observar las formas de onda de salida. Para cada conjunto de señales de salida, determinar si la tarjeta funciona correctamente, y si no es así, especificar los posibles fallos.

■ *Actividad 2.* Diseñar un método manual que se pueda emplear para probar la lógica de control.

EWB

La lógica de control se encuentra en el archivo SA05.EWB del CD-ROM EWB. Utilícese el generador y el analizador lógico para simular esta prueba para cada uno de los fallos.

Tarjeta 3 Analizador lógico

Tarjeta 2 Analizador lógico

Tarjeta 1 Analizador lógico

Tarjeta 4 Analizador lógico

M_3 M_2 M_1 M_0 S_3 S_2 S_1 S_0

Contador binario

Impulsos de reloj (CLK)

FIGURA P.5
Páginas representativas de una sección típica dedicada a las aplicaciones de sistemas digitales.

Contenido y organización

Los quince capítulos de *Fundamentos de sistemas digitales* cubren un amplio rango de temas, comenzando con los conceptos digitales básicos y siguiendo con los sistemas de numeración, las puertas lógicas, el álgebra booleana, la lógica combinacional, una introducción a los dispositivos PLD, la lógica secuencial (incluyendo su implementación con dispositivos PLD), las memorias y el almacenamiento, las interfaces, una introducción básica a los microprocesadores y computadoras y las tecnologías de circuitos integrados. Puede ser conveniente realizar una exposición más detallada de alguna de estas áreas.

Tecnologías de circuitos integrados. El Capítulo 15 es único en cuanto a que puede emplearse como un "capítulo flotante", que puede estudiarse parcial o completamente en cualquier punto del texto, o puede omitirse por completo sin que ello afecte al resto de los temas. Se han incluido avisos a lo largo del texto para sugerir los puntos en los que pueden leerse temas de dicho capítulo, aunque éste puede abordarse en cualquier punto posterior al Capítulo 3. Su ubicación como último capítulo tiene la intención de facilitar su uso flexible y el diseño de los bordes de sus páginas proporciona una referencia sencilla y rápida. También, aunque este Capítulo 15 se omita, el libro puede abordarse sin que sea necesario que el estudiante tenga conocimientos sobre los circuitos de transistores. El Capítulo 3 proporciona información suficiente sobre los circuitos integrados digitales para aquéllos que no tengan conocimientos sobre circuitos de transistores. Para aquéllos que deseen conocer los detalles de la tecnología de circuitos integrados, el Capítulo 15 proporciona dicha información.

Dispositivos lógicos programables. Los Capítulos 7 y 11 hacen una introducción al importante tema de los dispositivos lógicos programables (PLD) y, si se desea, pueden tratarse como opcionales. Cada uno de ellos, o ambos, se pueden omitir sin que los demás temas se vean afectados. El Capítulo 7 continúa con la exposición de la lógica combinacional y proporciona una introducción a los dispositivos PLD y a la programación de los mismos, en lo relacionado con la lógica combinacional, exclusivamente. El Capítulo 11 sigue al tema de la lógica secuencial y continúa con la exposición sobre PLD del Capítulo 7, incluyendo una introducción a la implementación de la lógica secuencial utilizando estos dispositivos. Hay disponibles una serie de lenguajes de programación para PLD, siendo ABEL el que se utiliza en este texto para ilustrar la programación de dispositivos PLD . Es posible que algunos profesores prefieran retrasar el estudio del Capítulo 7 hasta haber visto el Capítulo 10, con el fin de ver los capítulos sobre dispositivos PLD seguidos.

Microprocesadores. El Capítulo 14 proporciona una breve introducción a los microprocesadores y computadoras. Se exponen tanto los aspectos hardware como software. Se utiliza la familia de microprocesadores 8086/8088 de Intel como modelo básico, para introducir conceptos que son válidos incluso para los microprocesadores más recientes. El capítulo cubre las funcionalidades de todos los dispositivos de Intel hasta el Pentium e introduce la familia de microprocesadores de Motorola. Por supuesto, este capítulo no pretende cubrir completamente el tema, ya que hacerlo ocuparía uno o más libros de texto completos. Sin embargo, puede utilizarse como introducción a los microprocesadores para los estudiantes que más adelante vayan a seguir un curso dedicado a dicho tema.

Últimas materias. Al final del libro se encuentra una serie de apéndices, que contienen varias hojas de características representativas, una tabla de conversiones de código y una tabla de las potencias de dos. Los apéndices van seguidos de las respuestas a los problemas impares seleccionados. También se incluye un exhaustivo glosario que contiene las definiciones de las palabras clave de cada capítulo, además de otras definiciones. Por último, detrás del glosario se incluye el índice alfabético.

Sugerencias de uso de *Fundamentos de los sistemas digitales*

Generalmente, las limitaciones de tiempo o la profundidad del curso restringen los temas que se pueden cubrir en un trimestre. También, es habitual alterar la secuencia de ciertos temas con respecto al orden que siguen en este texto. Las siguientes sugerencias para realizar un tratamiento más ligero o selectivo para omitir determinados temas, no implican que un determinado tema sea menos importante que otros, sino que, dentro del contexto de un programa específico, hay temas en los que no es necesario hacer tanto énfasis como en los temas fundamentales. Estas sugerencias no necesariamente reflejan todas las posibilidades de alteración de la secuencia, ni de una cobertura más ligera o selectiva, ni tampoco de lo que se pueda omitir. En cualquier otro tipo de curso, pueden existir otras áreas que tener en cuenta.

Sugerencias para alterar la secuencia de los capítulos:

1. Si desea cubrir las puertas lógicas al principio del curso, puede tratar el Capítulo 1 por encima y el Capítulo 3 dedicado a las puertas lógicas puede verse antes que el Capítulo 2, que se ocupa de los sistemas de numeración, operaciones y códigos.
2. El Capítulo 7 sobre dispositivos PLD puede posponerse para estudiarse después del Capítulo 10. De este modo, los dos capítulos (7 y 11) que tratan estos dispositivos pueden abordarse de forma consecutiva, si así el curso es más efectivo.
3. El Capítulo 15 puede verse en cualquier punto del texto. Por ejemplo, puede estudiarse parcial o completamente después del Capítulo 3.

Capítulos que pueden considerarse para un cobertura selectiva

1. Capítulo 1, Introducción a los conceptos digitales.
2. Capítulo 2, Sistemas de numeración, operaciones y códigos.
3. Capítulo 4, Álgebra de Boole y simplificación lógica.
4. Capítulo 13, Interfaces.
5. Capítulo 15, Tecnologías de circuitos integrados.

Capítulos que pueden omitirse sin que afecten a otros temas

1. Capítulo 7, Introducción a los dispositivos lógicos programables. Debido a la creciente importancia de este tema, debería pensárselo dos veces antes de omitir este capítulo o el Capítulo 11.
2. Capítulo 11, Aplicaciones de los dispositivos PLD a la lógica secuencial.
3. Capítulo 13, Interfaces.
4. Capítulo 14, Introducción a los microprocesadores y computadoras.
5. Capítulo 15, Tecnologías de circuitos integrados.

Dependiendo del tipo de curso, pueden tratarse de forma ligera u omitirse algunos otros temas adicionales. Por ejemplo, las secciones dedicadas a las aplicaciones de los sistemas digitales se pueden omitir sin que ello afecte a otros temas, o pueden asignarse para la obtención de créditos adicionales o como proyectos especiales.

Agradecimientos

Muchas personas de gran capacidad han colaborado en esta séptima edición de *Fundamentos de sistemas digitales*. El libro ha sido completamente revisado, tanto en

lo que se refiere a contenido cuanto en términos de precisión. Entre las personas de Prentice Hall que han resultado cruciales durante el proceso de producción de este libro podemos citar a Rex Davidson, Katie Bradford y Scott Sambucci. Se merecen un gran reconocimiento. Mi aprecio, una vez más, a Lois Porter, que ha vuelto a hacer un maravilloso trabajo editando el manuscrito. Gracias también a Jane López por su gran labor en lo relativo a los gráficos usados en el texto. Gary Snyder ha sido de nuevo el encargado de verificar la precisión del texto, y deseo felicitarle por el sobresaliente trabajo realizado. Además, Gary ha creado los archivos de circuito EWB para los ejercicios del banco de trabajo electrónico que se han introducido en esta edición. Por último, y no por ello menos importante, gracias a mi colega Dave Buchla por su contribución a la revisión del Capítulo 14, sus sugerencias sobre numerosos temas y su cuidadosa revisión del manuscrito.

Como siempre, el éxito de un texto como éste depende de los expertos comentarios de muchos usuarios y lectores de *Fundamentos de sistemas digitales.* Vaya mi agradecimiento a los siguientes revisores, que han realizado muchas valiosas sugerencias y ejercido una constructiva crítica.

William H. Brownlowe, Montgomery County Community College

Hikmat Chedid, Lorain County Community College

John P. Corgan, Luzenne County Community College

Gene Craft, Lima Technical College

Lance Crimm, Southern Polytechnic State University

Paul A. Dilsner, Seminole Community College

Floyd Martin, Santa Ana College

Keith Quigley, Midlans Technical College

Nelson A. Riedel, Central Ohio Technical College

Tom Floyd

DEDICATORIA

A DEBBIE Y CYNDI

1

INTRODUCCIÓN A LOS CONCEPTOS DIGITALES

1.1 Magnitudes analógicas y digitales
1.2 Dígitos binarios, niveles lógicos y formas de onda digitales
1.3 Operaciones lógicas básicas
1.4 Funciones lógicas básicas
1.5 Circuitos digitales integrados
1.6 Instrumentación para pruebas y localización de averías
1.7 Aplicación a los sistemas digitales

Simulación por computadora de circuitos digitales:
El tutorial de Electronics Workbench (EWB) se encuentra en la dirección http://www.prenhall.com/floyd

■ OBJETIVOS DEL CAPÍTULO

- ❑ Explicar las diferencias básicas entre las magnitudes digitales y analógicas.
- ❑ Mostrar cómo se usan los niveles de tensión para representar magnitudes digitales.
- ❑ Describir los diferentes parámetros de los trenes de impulsos, tales como tiempo de subida, tiempo de bajada, ancho del impulso, frecuencia, periodo y ciclo de trabajo.
- ❑ Explicar las operaciones lógicas básicas NOT, AND y OR.
- ❑ Describir las funciones básicas del comparador, sumador, convertidor de código, codificador, decodificador, multiplexor, demultiplexor, contador y registro.
- ❑ Identificar los circuitos integrados digitales de acuerdo con su complejidad y el tipo de encapsulado.
- ❑ Determinar la numeración de los pines en los encapsulados de los circuitos integrados.
- ❑ Reconocer los instrumentos digitales y entender cómo se usan en la localización de averías en circuitos y sistemas digitales.
- ❑ Mostrar en una aplicación práctica cómo se forma un sistema digital completo a partir de las funciones básicas.

■ INTRODUCCIÓN

El concepto de computadora digital se remonta a Charles Babbage, quien desarrolló un basto dispositivo de computación mecánico hacia 1830. La primera computadora digital funcional fue construida en 1944 en la Universidad de Harvard, pero era electromecánica, no electrónica. La electrónica digital moderna comenzó en 1946 con una computadora digital electrónica llamada ENIAC, que fue fabricada con válvulas de vacío. Aunque ocupaba una habitación entera, ENIAC no tenía ni siquiera la potencia que puede tener hoy en día una calculadora de bolsillo.

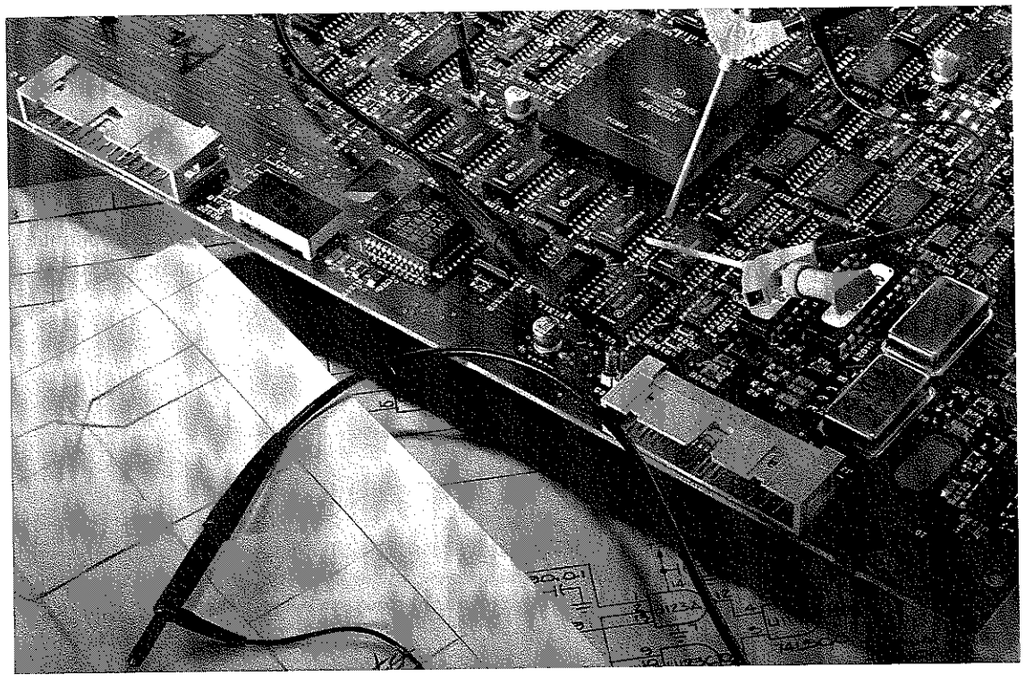
El término *digital* se deriva de la forma en que las computadoras realizan las operaciones: contando dígitos. Durante muchos años, las aplicaciones de la electrónica digital se limitaron a los sistemas informáticos. Hoy en día, la tecnología digital tiene aplicación en una amplia variedad de áreas. Aplicaciones tales como la televisión, sistemas de comunicaciones, de radar, sistemas de navegación y guiado, sistemas militares, instrumentación médica, control de procesos industriales y electrónica de consumo, usan todos ellos técnicas digitales. La tecnología digital ha progresado desde los circuitos de válvulas de vacío hasta los transistores discretos y los circuitos integrados complejos, conteniendo algunos de ellos millones de transistores.

■ APLICACIÓN A LOS SISTEMAS DIGITALES

La última sección de la mayor parte de los capítulos de este libro se dedica a una aplicación, que resume los principales aspectos abordados en el capítulo. Cada sistema está diseñado para ilustrar, en cada capítulo, cómo pueden utilizarse la teoría y los dispositivos. A lo largo del libro se presentan cinco sistemas diferentes, abarcando alguno de ellos dos o más capítulos.

Todos los sistemas se han simplificado para hacerlos manejables dentro del contexto del tema del capítulo. Aunque están basados en los requisitos reales del sistema, están diseñados para adecuarse al contenido del capítulo y no pretenden representar el método más eficiente y más moderno para dicha aplicación.

La sección 1.7 presenta el primer sistema, que es un sistema de control industrial para contar y controlar los objetos para envasado sobre una cinta transportadora. Está diseñado para incorporar todas las funciones lógicas que se introducen en este capítulo, de forma que se pueda ver cómo se utilizan y cómo interaccionan para conseguir un objetivo global.



Copyright 1998 Tektronix, Inc. Todos los derechos reservados. Reproducida con su permiso.

1.1 ■ MAGNITUDES ANALÓGICAS Y DIGITALES

Los circuitos electrónicos se pueden dividir en dos amplias categorías: digitales y analógicos. La electrónica digital utiliza magnitudes con valores discretos, mientras que la electrónica analógica emplea magnitudes con valores continuos. Aunque en este libro vamos a estudiar los fundamentos digitales, también debemos conocer los analógicos, dado que muchas aplicaciones requieren la utilización de ambos.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Definir el término *analógico*. ■ Definir el término *digital*. ■ Explicar las diferencias entre magnitudes digitales y analógicas. ■ Establecer las ventajas de digital frente a analógico. ■ Dar ejemplos sobre cómo se utilizan en electrónica las magnitudes digitales y analógicas.

Una magnitud **analógica*** es aquella que toma valores continuos. Una magnitud **digital** es aquella que toma un conjunto de valores discretos. La mayoría de las cosas que se puedan medir cuantitativamente aparecen en la naturaleza en forma analógica. Por ejemplo, la temperatura varía dentro de un rango continuo de valores. A lo largo de un día, la temperatura no varía entre, por ejemplo, 20°C y 25°C de forma instantánea, sino que alcanza todos los infinitos valores que hay en ese intervalo. Si dibujamos la gráfica de temperatura de un día típico de verano, tendríamos una curva continua suave similar a la de la Figura 1.1. Otros ejemplos de magnitudes analógicas son el tiempo, la presión, la distancia o el sonido.

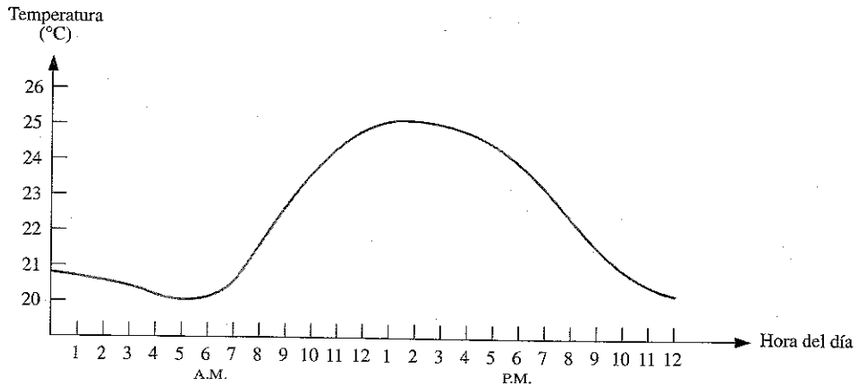


FIGURA 1.1
Gráfica de una magnitud analógica (temperatura en función del tiempo).

Mejor que hacer una gráfica de temperatura en un rango continuo, supongamos que simplemente medimos la temperatura cada hora. Lo que tenemos ahora son

* Las definiciones de los términos en negrita se encuentran en el glosario al final del libro. También se han seleccionado algunos términos para incluirlos en la sección "Palabras clave" al final de cada capítulo.

muestras que representan la temperatura a lo largo de intervalos discretos de tiempo, sobre un periodo de 24 horas, tal y como se indica en la Figura 1.2. De esta forma, hemos convertido una magnitud analógica a un formato que se puede ahora digitalizar, representando cada valor muestreado mediante un código digital. Es importante darse cuenta de que la Figura 1.2 no es la representación digital de la magnitud analógica.

La ventaja de las magnitudes digitales. En las aplicaciones electrónicas, las magnitudes digitales tienen ciertas ventajas sobre las analógicas. La principal ventaja es que los datos digitales se pueden procesar y transmitir de forma más eficiente y fiable que los datos analógicos. También poseen otra gran ventaja cuando es necesario proceder a su almacenamiento. Por ejemplo, cuando la música se convierte a formato digital, puede almacenarse de manera más compacta y reproducirse con mayor precisión y claridad de lo que es posible en formato analógico.

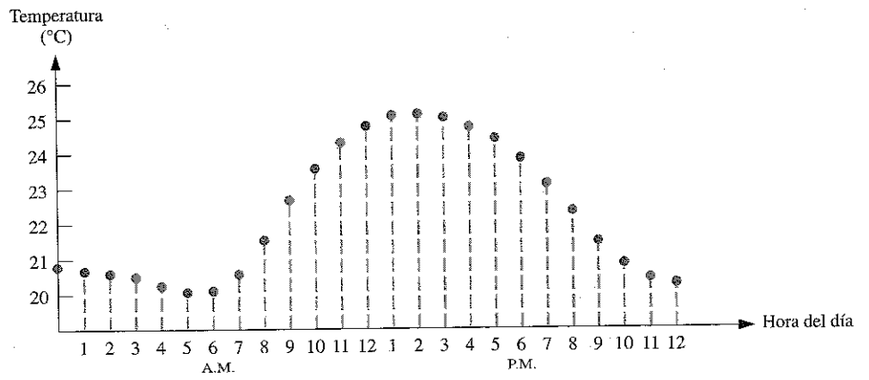


FIGURA 1.2
Representación de los valores muestreados (cuantificación) de la magnitud analógica de la Figura 1.1. Cada valor representado por un punto puede digitalizarse, representándolo como un código digital que consta de una serie de 1s y 0s.

El ruido (fluctuaciones de tensión no deseadas) no afecta a los datos digitales tanto como a las señales analógicas.

Un sistema electrónico analógico

Un sistema de altavoz, que se emplea para amplificar el sonido de forma que éste sea oído por una gran audiencia, es un ejemplo de una aplicación de electrónica analógica. El diagrama básico de la Figura 1.3 ilustra cómo estas ondas de sonido, que son analógicas en su origen, son capturadas por un micrófono y convertidas en una pequeña variación analógica de tensión denominada señal de audio. Esta tensión varía de manera continua a medida que cambia el volumen y la frecuencia del sonido, y se aplica a la entrada de un amplificador lineal. La salida del amplificador, que es la tensión

de entrada amplificada, se introduce en el altavoz. Éste convierte, de nuevo, la señal de audio amplificada en ondas sonoras con un volumen mucho mayor que el sonido original captado por el micrófono.

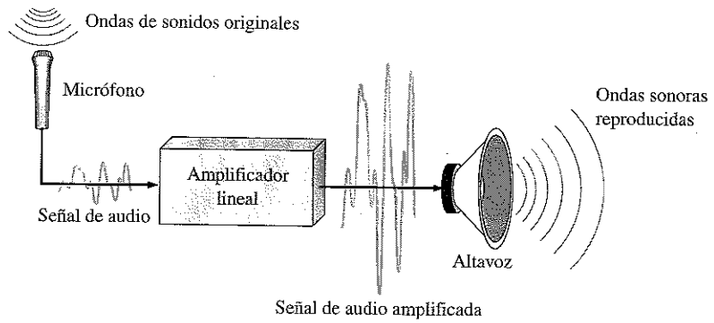


FIGURA 1.3
Sistema básico de altavoz.

Sistema que utiliza métodos digital y analógico

Un reproductor de disco compact (CD) es un ejemplo de sistema en el que se utilizan circuitos analógicos y digitales. El diagrama simplificado de la Figura 1.4 ilustra el principio básico. La música en formato digital se almacena en el CD. Un sistema óptico de diodos láser lee los datos digitales del disco cuando éste gira y los transfiere al **convertidor digital-analógico (DAC,** digital-to-analog converter). El DAC transforma los datos digitales en una señal analógica que es la reproducción eléctrica de la música original. Esta señal se amplifica y se envía al altavoz para que podamos disfrutarla. Cuando la música original se grabó en el CD se utilizó un proceso que, esencialmente, era el inverso del descrito aquí, y que utilizaba un **convertidor analógico-digital (ADC,** analog-to-digital converter).

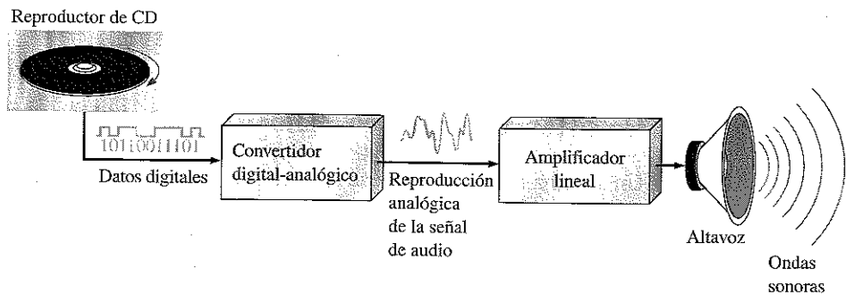


FIGURA 1.4
Esquema básico de un reproductor de CD. Sólo se muestra un canal.

REVISIÓN DE LA SECCIÓN 1.1

Respuestas al final del capítulo.

1. Definir *analógico*.
2. Definir *digital*.
3. Explicar la diferencia entre una magnitud digital y una magnitud analógica.
4. Dar un ejemplo de un sistema que sea analógico y de uno que sea una combinación de digital y analógico. Nombrar un sistema que sea enteramente digital.

1.2 ■ DÍGITOS BINARIOS, NIVELES LÓGICOS Y FORMAS DE ONDA DIGITALES

La electrónica digital utiliza sistemas y circuitos en los que sólo existen dos estados posibles. Estos estados se representan mediante dos niveles de tensión diferentes: ALTO (HIGH) y BAJO (LOW). Estos dos estados pueden representarse también mediante niveles de corriente, interruptores abiertos o cerrados, o lámparas encendidas o apagadas. En los sistemas digitales, las combinaciones de estos dos estados se denominan códigos, y se utilizan para representar números, símbolos, caracteres alfabéticos y cualquier otro tipo de información. El sistema de numeración de dos estados se denomina binario y los dos dígitos que emplea son 1 y 0. Un dígito binario se denomina bit.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Definir *binario*. ■ Definir *bit*. ■ Nombrar los bits en un sistema binario. ■ Explicar cómo se emplean los niveles de tensión para representar bits. ■ Explicar cómo interpretan los circuitos digitales los niveles de tensión. ■ Describir las características generales de un impulso. ■ Determinar la amplitud, tiempo de subida, tiempo de bajada y ancho de un impulso. ■ Identificar y describir las características de una señal digital. ■ Determinar la amplitud, periodo, frecuencia y ciclo de trabajo de una señal digital. ■ Explicar qué es un cronograma o diagrama de tiempos y establecer su finalidad. ■ Explicar la transferencia de datos paralelo y serie, y las ventajas e inconvenientes de ambos métodos.

Dígitos binarios

Los dos dígitos del sistema **binario**, 1 y 0, se denominan **bits**, que es la contracción de las palabras *binary digit* (dígito binario). En los circuitos digitales, se emplean dos niveles de tensión distintos para representar los dos bits. Un 1 se representa mediante un nivel de tensión más elevado, que se denomina nivel ALTO (HIGH), y un 0 se representa mediante un nivel más bajo de tensión, que se denomina BAJO (LOW). Este convenio recibe el nombre de **lógica positiva**, y es la que se va a emplear a lo largo del libro.

ALTO (HIGH) = 1 y BAJO (LOW) = 0

Al sistema, mucho menos común, en el que un 1 se representa por un nivel BAJO y un 0 por un nivel ALTO se le denomina *lógica negativa*.

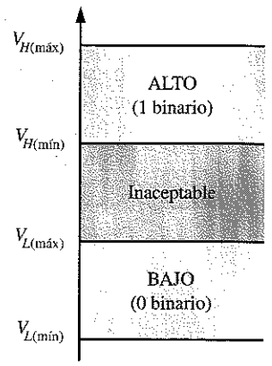
Los grupos de bits (combinaciones de 1s y 0s), llamados *códigos*, se emplean para representar números, letras, símbolos, instrucciones y cualquier otra cosa que se requiera en una cierta aplicación.

Niveles lógicos

Las tensiones que se utilizan para representar los unos y ceros reciben el nombre de niveles *lógicos*. Lo ideal sería que un nivel de tensión representara el nivel ALTO (HIGH) y otro nivel de tensión representara el nivel BAJO (LOW). Sin embargo, en un circuito digital práctico, un nivel ALTO puede ser cualquier tensión entre un máximo y un mínimo especificados. De igual manera, un nivel BAJO puede ser cualquier tensión comprendida entre un máximo y un mínimo especificados. No puede existir solapamiento entre los niveles ALTOS aceptados y los niveles BAJOS aceptados.

La Figura 1.5 ilustra el rango general de variación para los niveles BAJO y ALTO en un circuito digital. La variable $V_{H(máx)}$ representa el máximo valor para el nivel ALTO y $V_{H(mín)}$ representa el mínimo valor para el nivel ALTO. El valor máximo del nivel BAJO se representa mediante $V_{L(máx)}$ y su valor mínimo por $V_{L(mín)}$. Los valores de tensión comprendidos entre $V_{L(máx)}$ y $V_{H(mín)}$ no son aceptables para un funcionamiento correcto. Una tensión dentro de este rango podría ser interpretada tanto como nivel ALTO cuanto como nivel BAJO en un determinado circuito. Por tanto, estos valores no aceptables no deben emplearse nunca. Por ejemplo, los valores del nivel ALTO para un cierto tipo de circuito digital denominado TTL pueden variar desde 2 V a 5 V mientras que los valores del nivel BAJO lo pueden hacer entre 0 V y 0,8 V. De esta manera, si por ejemplo se aplica una tensión de 3,5 V, el circuito lo interpretará como un nivel ALTO (HIGH) o 1 binario. Si se aplica una tensión de 0,5 V, el circuito lo interpretará como un nivel BAJO (LOW) o 0 binario. Para este tipo de circuito, las tensiones comprendidas entre 0,8 V y 2 V no son aceptables y nunca deben ser utilizadas.

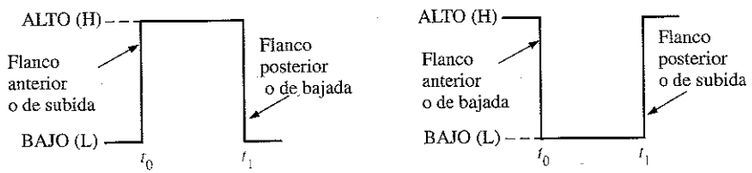
FIGURA 1.5
Rango de niveles lógicos de tensión para un circuito digital.



Formas de onda digitales

Las señales digitales consisten en niveles de tensión que varían entre los estados o niveles ALTO y BAJO. La Figura 1.6(a) muestra que un **impulso** positivo se genera cuando la tensión (o la intensidad) pasa de su nivel normalmente BAJO hasta su nivel ALTO, y luego otra vez retorna al nivel BAJO. El impulso negativo de la Figura 1.6(b) se genera cuando la tensión pasa de su nivel ALTO hasta el nivel BAJO, y vuelve otra vez al nivel ALTO. Una señal digital está compuesta por una serie de impulsos.

Impulsos. Como se indica en la Figura 1.6, un impulso posee dos flancos: un **flanco anterior**, que se produce en el instante t_0, y un **flanco posterior** que se produce en el instante t_1. Para un impulso positivo, el flanco anterior es un flanco de subida y el flanco posterior es de bajada. Los impulsos de la Figura 1.6 son ideales, ya que se supone que los flancos de subida y bajada ocurren en un tiempo cero (instantáneamente). En la práctica, estas transiciones no suceden de forma instantánea, aunque para la mayoría de las situaciones podemos asumir que son impulsos ideales.



(a) Impulso positivo (b) Impulso negativo

FIGURA 1.6
Impulsos ideales.

La Figura 1.7 muestra un impulso real (no ideal). El tiempo requerido por el impulso para pasar del nivel BAJO al nivel ALTO recibe el nombre de **tiempo de subida** (t_r), mientras que el tiempo requerido para la transición del nivel ALTO al nivel BAJO se denomina **tiempo de bajada** (t_f). En la práctica, el tiempo de subida se mide como el tiempo que tarda en pasar del 10% (distancia desde la línea base) al 90% de la **amplitud** del impulso, y el tiempo de bajada se mide como el tiempo que tarda en pasar del 90% al 10% de la amplitud del impulso, como se puede observar en la Figura 1.7. La razón de que el 10% inferior y el 10% superior no se incluyan dentro de los tiempos de subida y bajada se debe a la no linealidad de la señal en estas áreas. La **anchura del impulso** (t_w) es una medida de la duración del impulso y, a menudo, se define como el intervalo de tiempo que transcurre entre los puntos en los que la amplitud es del 50% en el flanco de subida y en el de bajada, como se indica en la Figura 1.7.

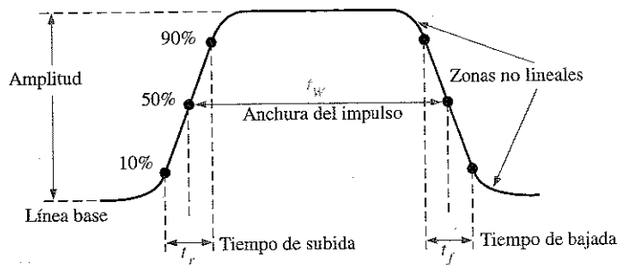


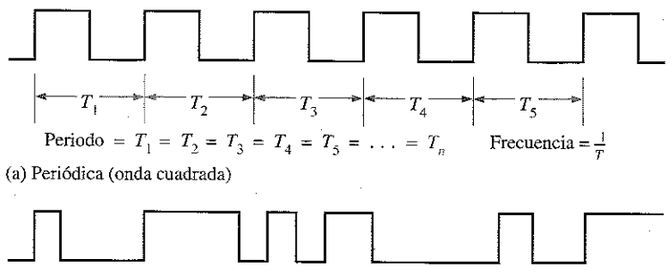
FIGURA 1.7
Características de un impulso no ideal.

Características de las formas de onda. La mayoría de las señales que podemos encontrar en los sistemas digitales se compone de series de impulsos, algunas veces llamados también *trenes de impulsos,* y pueden clasificarse en periódicas y no periódicas. Un tren de impulsos **periódico** es aquel que se repite a intervalos de tiempo fijos, este intervalo de tiempo fijo se denomina **periodo** (T). La frecuencia (f) es la velocidad a la que se repite y se mide en hertzios (Hz). Un tren de impulsos no periódico, claro está, no se repite a intervalos de tiempo fijos, y puede estar compuesto por impulsos de distintos anchos y/o impulsos que tienen intervalos diferentes de tiempo entre impulsos. Un ejemplo de cada tipo se muestra en la Figura 1.8.

La **frecuencia** (f) de un tren de impulsos (digital) es el inverso del periodo. Las relaciones entre frecuencia y periodo se pueden expresar de la siguiente manera:

$$f = \frac{1}{T} \tag{1.1}$$

$$T = \frac{1}{f} \tag{1.2}$$



(a) Periódica (onda cuadrada)

(b) No periódica

FIGURA 1.8
Ejemplos de formas de onda digitales.

Una característica importante de una señal digital periódica es su ciclo de trabajo. El **ciclo de trabajo** se define como la razón entre el ancho del impulso (t_w) y el periodo (T) y puede expresarse como un porcentaje.

$$\text{Ciclo de trabajo} = \left(\frac{t_w}{T}\right)100\% \tag{1.3}$$

EJEMPLO 1.1 En la Figura 1.9 se muestra una parte de una señal digital periódica. Las medidas están expresadas en milisegundos. Determinar:

(a) periodo **(b)** frecuencia **(c)** ciclo de trabajo

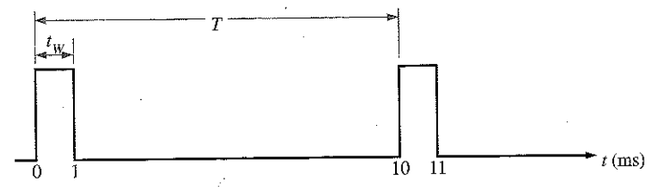


FIGURA 1.9

Solución

(a) El periodo se mide desde el flanco de uno de los impulsos hasta el correspondiente flanco del impulso siguiente. En este caso, T se mide desde el flanco de subida hasta el siguiente flanco de subida, como se indica en la figura, luego T es igual a **10 ms**.

(b) $f = \frac{1}{T} = \frac{1}{10\text{ ms}} = \mathbf{100\ Hz}$

(c) Ciclo de trabajo $= \left(\frac{t_W}{T}\right)100\% = \left(\frac{1\text{ ms}}{10\text{ ms}}\right)100\% = \mathbf{10\%}$

Problema relacionado*. Una señal digital tiene un ancho de impulso de 25 µs y un periodo de 150 µs. Calcular la frecuencia y el ciclo de trabajo.

NOTAS INFORMÁTICAS
La velocidad a la que una computadora puede funcionar depende del tipo de microprocesador utilizado en el sistema. La especificación de velocidad de una computadora de, por ejemplo, 300 MHz, es la frecuencia máxima de reloj a la que el microprocesador puede trabajar.

Una señal digital contiene información binaria

La información binaria que manejan los sistemas digitales aparece en forma de señales que representan secuencias de bits. Cuando la señal está a nivel ALTO, se representa con un 1 binario, mientras que si la señal está a nivel BAJO, lo indica un 0 binario. Cada bit dentro de una secuencia ocupa un intervalo de tiempo definido, denominado **periodo del bit**.

El reloj. En los sistemas digitales, todas las señales se sincronizan con una señal de temporización básica denominada **reloj**. El reloj es una señal periódica en la que cada intervalo entre impulsos (el periodo) equivale a la duración del bit.

Un ejemplo de una señal de reloj se muestra en la Figura 1.10. Obsérvese que, en este caso, cada cambio de nivel de la señal A se produce en el flanco de subida de la señal de reloj. En otros casos, el cambio de nivel se produce en el flanco de bajada de dicha señal. Para cada impulso de la señal de reloj, la señal A puede estar a nivel ALTO o BAJO. Como hemos dicho, estos niveles de tensión altos o bajos representan una secuencia de bits. Un grupo de varios bits puede usarse como parte de una información binaria, igual que un número o una letra. La señal de reloj en sí misma no lleva información.

Cronogramas o diagramas de tiempos

Un **diagrama de tiempos** es una gráfica de formas de onda digitales que muestra la relación temporal real entre dos o más señales, y cómo varía cada señal en relación

* Las respuestas a todos los problemas relacionados se encuentran al final del capítulo.

con las demás. La Figura 1.10 es un ejemplo de un sencillo diagrama de tiempos que muestra cómo se relacionan la señal de reloj y la señal *A* en el tiempo.

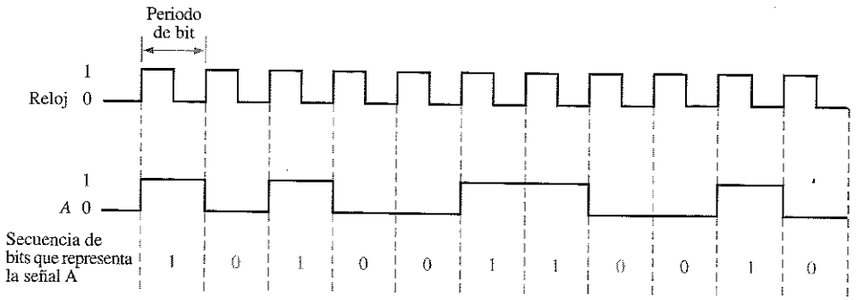


FIGURA 1.10
Ejemplo de una señal de reloj sincronizada con una señal que representa una secuencia de bits.

Examinando un diagrama de tiempos, se pueden determinar los estados (ALTO o BAJO), de cada una de las señales en cualquier instante de tiempo especificado, y el instante exacto en que cualquiera de las señales cambia de estado con respecto a las restantes. La Figura 1.11 es un ejemplo de un cronograma con cuatro señales. A partir de este diagrama podemos ver, por ejemplo, que las tres señales (*A*, *B* y *C*) están a nivel ALTO sólo durante el séptimo ciclo de reloj, y las tres pasan de nuevo a nivel BAJO cuando termina este mismo ciclo.

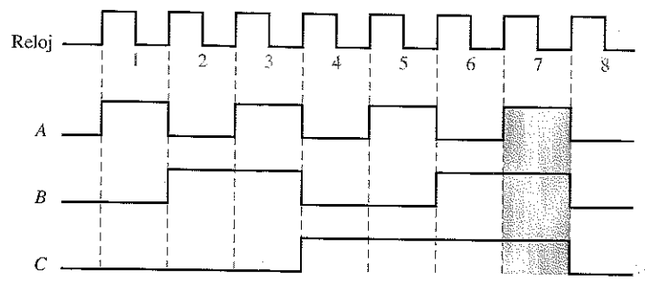
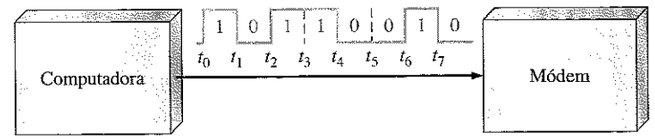


FIGURA 1.11
Ejemplo de un cronograma o diagrama de tiempos.

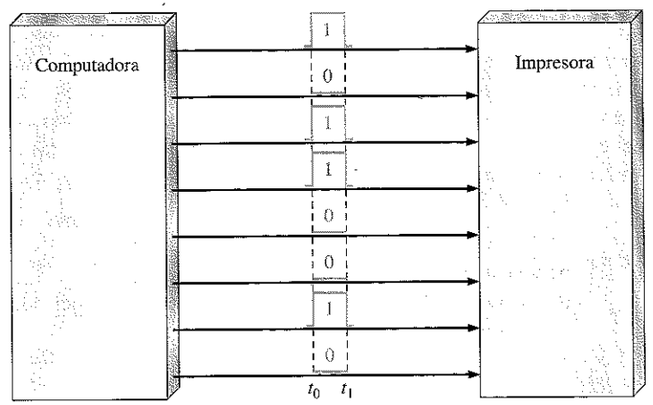
Transferencia de datos

Por **datos** entendemos grupos de bits que transportan algún tipo de información. Los datos binarios, que se representan mediante códigos digitales, deben transferirse de un circuito a otro dentro de un mismo sistema digital, o desde un sistema a otro, para poder llevar a cabo una determinada función. Por ejemplo, los números almacenados

en formato binario en la memoria de una computadora deben transferirse a la unidad central de proceso para poder sumarse. Después, el resultado de esta suma tiene que llegar a la pantalla para visualizarse y/o transferirse de nuevo a la memoria. En los sistemas informáticos, como los mostrados en la Figura 1.12, los datos binarios se transfieren de dos formas: en serie y en paralelo.



(a) Transferencia en serie de datos binarios desde una computadora a un módem. El primer intervalo es t_0-t_1



(b) Transferencia en paralelo de datos binarios desde una computadora a una impresora. t_0 es el instante inicial.

FIGURA 1.12
Transferencia en serie y en paralelo de datos binarios. Sólo se muestran las líneas de datos.

NOTAS INFORMÁTICAS
Los datos serie se transmiten a través de un puerto serie de una computadora, conocido también como puerto COM. Los usos más comunes para un puerto serie son el ratón o un módem. Los datos en paralelo se transfieren a través del puerto paralelo de una computadora, conocido también como puerto LPT. Los usos más comunes de un puerto paralelo son con una impresora o una unidad de disco.

Cuando los bits se transmiten en **serie** desde un punto a otro, se envían bit a bit a lo largo de un único conductor, como se ilustra en la Figura 1.12(a), para el caso de una transmisión computadora-módem. En el intervalo de t_0 a t_1 se transmite el primer bit; en el intervalo t_1 a t_2 se transmite el segundo bit, y así sucesivamente. Luego transmitir ocho bits en serie precisa ocho intervalos de tiempo.

Cuando los bits se transmiten en **paralelo**, todos los bits del grupo se envían al tiempo por líneas separadas. Como se muestra en la Figura 1.12(b), para el caso en que se transmiten ocho bits desde una computadora a una impresora, existe una línea para cada bit. Para realizar la transmisión de ocho bits en paralelo se precisa solamente un intervalo de tiempo, frente a los ocho que se necesitan en la transmisión serie.

En resumen, la ventaja de una transmisión en serie de datos binarios es que sólo se necesita una línea. En la transmisión en paralelo se necesitan tantas líneas

como número de bits que hay que transmitir al mismo tiempo. La desventaja de la transmisión en serie es que necesita un intervalo de tiempo mayor que la transmisión en paralelo. Por ejemplo, si un bit puede transmitirse en 1 μs, entonces para transmitir 8 bits en serie se necesitan 8 μs, pero solamente 1 μs para hacerlo en paralelo. La desventaja de la transmisión en paralelo es que se precisan más líneas.

EJEMPLO 1.2

(a) Determinar el tiempo total necesario para transmitir los ocho bits de la señal *A* de la Figura 1.13, e indicar la secuencia de bits. El bit más a la izquierda es el primero que se transmite. Como referencia se emplea una señal de reloj a 100 kHz.

(b) ¿Cuál es el tiempo total necesario para transmitir los mismos ocho bits en paralelo?

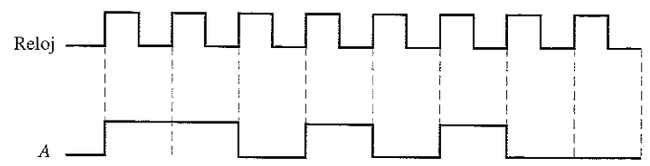


FIGURA 1.13

Solución

(a) Dado que la frecuencia de reloj es 100 kHz, el periodo es

$$T = \frac{1}{f} = \frac{1}{100\,\text{kHz}} = 10\mu\text{s}$$

Se necesitan 10 µs para transmitir cada bit de la señal. El tiempo total de transmisión es:

$$8 \times 10\ \mu\text{s} = \mathbf{80\ \mu s}$$

Para determinar la secuencia de bits, examinamos la señal de la Figura 1.13 para cada periodo de bit. Si la señal *A* está a nivel ALTO durante el periodo de bit, se transmite un 1. Si la señal *A* está a nivel BAJO durante dicho periodo, se transmite un 0. La secuencia de bits se ilustra en la Figura 1.14. El bit más a la izquierda es el primero que se transmite.

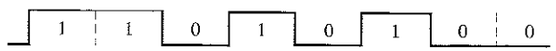


FIGURA 1.14

(b) Una transmisión en paralelo necesitaría **10 μs** para los ocho bits.

Problema relacionado. Si los datos binarios se transmiten a una velocidad de 10 millones de bits por segundo (10 Mbits/s), ¿cuánto tiempo será necesario para transmitir en paralelo 16 bits por 16 líneas? ¿cuánto se tardaría en transmitir los 16 bits en serie?

REVISIÓN DE LA SECCIÓN 1.2

1. Definir *binario*.
2. ¿Qué significa *bit*?
3. ¿Qué son los bits en un sistema binario?
4. ¿Cómo se miden el tiempo de subida y de bajada de un impulso?
5. Conociendo el periodo de una señal, ¿cómo se calcula su frecuencia?
6. Explicar qué es una señal de reloj.
7. ¿Cuál es el propósito de un cronograma o diagrama de tiempos?
8. ¿Cuál es la principal ventaja de la transmisión en paralelo de datos binarios frente a la transmisión en serie?

1.3 ■ OPERACIONES LÓGICAS BÁSICAS

En su forma más simple, la lógica es la parte del razonamiento humano que nos dice que una determinada proposición (sentencia de asignación) es cierta si se cumplen ciertas condiciones. Las proposiciones pueden ser clasificadas como verdaderas o como falsas. Muchas situaciones y procesos que encontramos en nuestra vida cotidiana pueden expresarse como funciones proposicionales o lógicas. Dado que tales funciones son sentencias verdaderas/falsas o afirmativas/negativas, pueden aplicarse a los circuitos digitales, ya que éstos se caracterizan por sus dos estados.

Al finalizar esta sección, el lector deberá ser capaz de:

- Enumerar las tres operaciones lógicas básicas.
- Definir la operación NOT.
- Definir la operación AND.
- Definir la operación OR.

Cuando se combinan varias proposiciones se forman funciones lógicas o proposicionales. Por ejemplo, la proposición "la luz está encendida" será cierta si "la bombilla no está fundida" lo es y si "el interruptor está dado" también es cierta. Por tanto, esta proposición lógica puede ser formulada como: *la luz está encendida sólo si la bombilla no está fundida y el interruptor está dado*. En este ejemplo, la primera afirmación es cierta sólo si las dos últimas lo son. La primera proposición ("la luz está encendida") es por tanto la proposición básica y las otras dos sentencias son las condiciones de las que depende la proposición.

Hacia 1850, el matemático y lógico irlandés George Boole desarrolló un sistema matemático para formular proposiciones lógicas con símbolos, de manera que los problemas puedan ser escritos y resueltos de una forma similar al álgebra ordinaria. El álgebra de Boole, como se le conoce hoy en día, encuentra aplicaciones en el diseño y el análisis de los sistemas digitales, y se tratará en detalle en el Capítulo 4.

El término **lógico** se aplica a los circuitos digitales que se utilizan para implementar funciones lógicas. Existen varios tipos de **circuitos** lógicos que son los elementos básicos que constituyen los bloques sobre los que se construyen sistemas digitales más complejos, como puede ser una computadora. Ahora vamos a ocuparnos de estos elementos y vamos a estudiar sus funciones de forma muy general. En capítulos posteriores se tratarán estos circuitos en detalle.

En la Figura 1.15 se presentan los símbolos estándar distintivos de las tres operaciones lógicas básicas. Existen también otros símbolos para estas operaciones que

se presentarán en el Capítulo 3. Las líneas conectadas a cada símbolo son las **entradas (inputs)** y **salidas (outputs)**. Las entradas son las líneas situadas a la izquierda de cada símbolo y la salida es la línea de la derecha. Un circuito que realiza una operación lógica determinada (NOT, AND, OR) se llama **puerta** lógica. Las puertas AND y OR pueden tener cualquier número de entradas, como se indica mediante las líneas de puntos en la Figura 1.15.

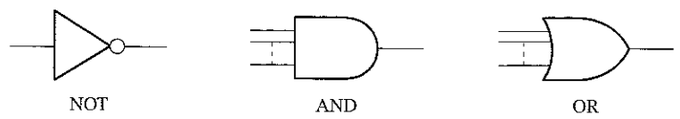


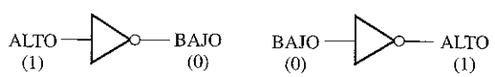
FIGURA 1.15
Operaciones lógicas básicas y sus símbolos.

En las operaciones lógicas, las condiciones mencionadas anteriormente de verdadero/falso se representan mediante un nivel ALTO (verdadero) y un nivel BAJO (falso). Cada una de las tres operaciones lógicas básicas da lugar a una única respuesta a un determinado conjunto de condiciones.

NOT

La operación **NOT** cambia de un nivel lógico al nivel lógico opuesto, como se indica en la Figura 1.16. Cuando la entrada está a nivel ALTO (1), la salida se pone a nivel BAJO (0). Cuando la entrada está a nivel BAJO, la salida se pone a nivel ALTO. En ambos casos, la salida *no* es la misma que la entrada. La operación NOT se implementa mediante un circuito lógico conocido como **inversor** o puerta NOT.

FIGURA 1.16
La operación NOT.



AND

La operación **AND** genera un nivel ALTO en la salida si y sólo si todas las entradas están a nivel ALTO, como puede verse en la Figura 1.17, para el caso de dos entradas. Cuando una de las entradas está a nivel ALTO y la otra entrada está a nivel ALTO, la salida se pone a nivel ALTO. Cuando cualquiera de las entradas está a nivel BAJO, la salida es un nivel BAJO. La operación AND se implementa mediante un circuito lógico conocido como *puerta AND*.

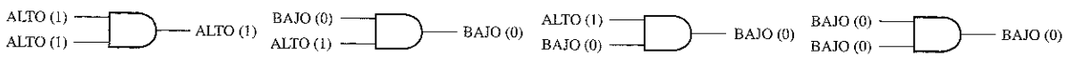


FIGURA 1.17
La operación AND.

OR

La operación **OR** genera un nivel ALTO de salida cuando cualquiera de sus entradas está a nivel ALTO, como se muestra en la Figura 1.18, para el caso de dos entradas. Cuando una de las dos entradas *o* ambas están a nivel ALTO, la salida se pone a nivel ALTO. Cuando las dos entradas están a nivel BAJO, la salida se pone a nivel BAJO. La operación OR se implementa mediante un circuito lógico denominado *puerta OR*.

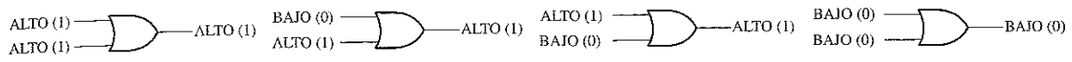


FIGURA 1.18
La operación OR.

REVISIÓN DE LA SECCIÓN 1.3

1. ¿Cuándo la operación NOT da lugar a una salida a nivel ALTO?
2. ¿Cuándo la operación AND da lugar a una salida a nivel ALTO?
3. ¿Cuándo la operación OR da lugar a una salida a nivel ALTO?
4. ¿Qué es un inversor?
5. ¿Qué es una puerta lógica?

1.4 ■ FUNCIONES LÓGICAS BÁSICAS

Los tres elementos lógicos básicos AND, OR y NOT pueden combinarse para formar circuitos lógicos más complejos, que realicen muchas operaciones útiles y que se empleen en la construcción de sistemas digitales completos. Algunas de las funciones lógicas más comunes son: comparación, aritmética, conversión de códigos, codificación, decodificación, selección de datos, almacenamiento y recuento. Esta sección proporciona una introducción general a estas funciones tan importantes, de tal manera que podamos empezar a ver cómo se forman los bloques básicos de sistemas digitales como la computadora. Cada una de estas funciones será tratada en detalle en los capítulos siguientes.

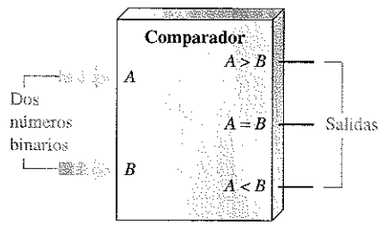
Al finalizar esta sección, el lector deberá ser capaz de:

■ Identificar ocho tipos básicos de funciones lógicas. ■ Describir un comparador de magnitud básico. ■ Enumerar las cuatro funciones aritméticas. ■ Describir un sumador básico. ■ Describir un codificador básico. ■ Describir un decodificador básico. ■ Definir multiplexación y demultiplexación. ■ Explicar cómo se realiza el almacenamiento de datos. ■ Describir la función de un contador básico.

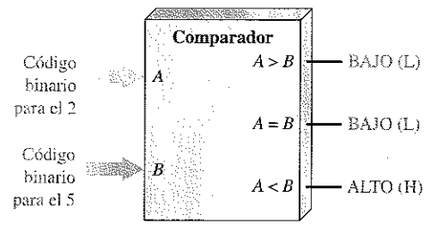
Función de comparación

La comparación de **magnitudes** se realiza mediante un circuito lógico denominado **comparador**, que se trata en el Capítulo 6. Su propósito es comparar dos cantidades

e indicar si son o no iguales. Por ejemplo, supongamos que tenemos dos números y que queremos saber si son o no iguales y, si son distintos, queremos saber cuál es el mayor. La función de comparación se representa en la Figura 1.19. Un número en formato binario se aplica a la entrada *A* y otro a la entrada *B*. Las salidas indican la relación entre los dos números, produciendo un nivel ALTO en la línea de salida apropiada. Supongamos que la representación binaria del número 2 se aplica a la entrada *A*, y a la entrada *B* se aplica la representación binaria del número 5 (en el siguiente capítulo nos ocuparemos de la representación binaria de los números y símbolos). En la salida *A*<*B* (*A* menor que *B*) aparecerá un nivel ALTO que indicará la relación entre ambos números (2 es menor que 5). Las flechas anchas indican un grupo de líneas paralelas a través de las cuales se transmiten los bits.



(a) Comparador de magnitud básico



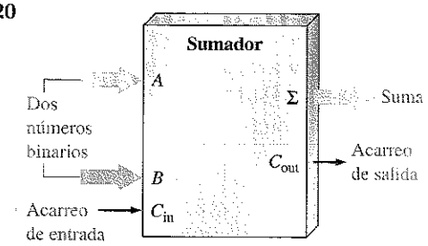
(b) Ejemplo: *A* menor que *B* (2<5) se indica mediante la salida a nivel ALTO (*A*<*B*)

FIGURA 1.19
La función de comparación.

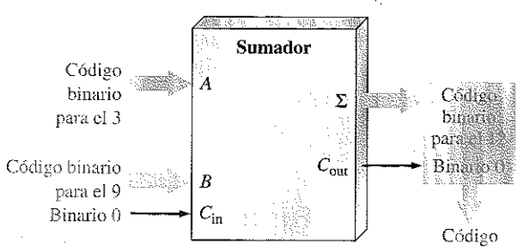
Funciones aritméticas

Suma. La adición se realiza mediante un circuito lógico denominado **sumador**, que se verá en el Capítulo 6. Su función es sumar dos números binarios (que se aplican a las entradas *A* y *B*, junto con una entrada de acarreo C_{in}), y generar la suma (Σ) y un acarreo de salida (C_{out}), como se muestra en la Figura 1.20(a). La Figura 1.20(b) ilustra la suma de los números 3 y 9. Sabemos que la suma es 12, este resultado se obtiene en las salidas del sumador obteniendo un 12 en código binario en la salida suma y un 0 en la salida de acarreo. Suponemos que la entrada de acarreo en este ejemplo está a 0.

FIGURA 1.20
La función suma.



(a) Sumador básico



(b) Ejemplo: *A* más *B* (3+9=12)

NOTAS INFORMÁTICAS
En un microprocesador, la unidad aritmético lógica (UAL o ALU) realiza las operaciones de suma, resta, multiplicación y división, así como las operaciones lógicas sobre los datos digitales mediante una serie de instrucciones. Una unidad aritmético lógica típica se construye con muchos miles de puertas lógicas.

Resta. La sustracción se realiza también mediante un circuito digital. Un **restador** requiere tres entradas: los dos números que han de ser restados y una entrada de acarreo negativo (*borrow*). Las dos salidas corresponden a la diferencia y a la salida de acarreo negativo. Cuando, por ejemplo, se resta 5 de 8 sin entrada de acarreo, la diferencia es 3 sin acarreo de salida. En el Capítulo 2 veremos cómo la sustracción se puede realizar mediante un sumador, ya que la sustracción es sólo un caso especial de la adición.

Multiplicación. La multiplicación se realiza mediante un circuito digital denominado *multiplicador*. Dado que los números se multiplican siempre de dos en dos, se necesitan dos entradas. La salida del multiplicador es el producto. Puesto que la multiplicación es sencillamente una serie de sumas con desplazamientos de posición de los productos parciales, se puede realizar empleando un sumador junto con otros circuitos.

División. La división se puede realizar mediante una serie de sustracciones, comparaciones y desplazamientos, por lo que también puede efectuarse utilizando un sumador junto con otros circuitos. En el divisor se necesitan dos entradas y las salidas que se generan son el cociente y el resto.

Función de conversión de código

Un **código** es un conjunto de bits ordenados de acuerdo a un único modelo y se emplean para representar una información específica. Un convertidor de código cambia el formato de una información codificada a otro formato de código. Ejemplos de estas conversiones son las conversiones entre binario y otros códigos como el código decimal binario (BCD, Binary Coded Decimal) o el código Gray. En el Capítulo 2 se verán distintos tipos de códigos, y en el Capítulo 6 se tratarán los convertidores de código.

Función de codificación

La función de codificación se implementa mediante un circuito denominado **codificador**, que se verá en el Capítulo 6. Un codificador convierte la información, tal y como puede ser un número decimal o un carácter del alfabeto, en algún tipo de código. Por ejemplo, un cierto tipo de codificador convierte los dígitos decimales, de 0 a 9, a código binario. Un nivel ALTO en la entrada correspondiente a un determinado dígito decimal genera el código binario apropiado en las líneas de salida.

La Figura 1.21 es una sencilla ilustración de un codificador que se usa para convertir (codificar) la pulsación de una tecla de la calculadora en un código binario, que puede ser procesado por los circuitos de la calculadora.

Función de decodificación

La función de decodificación se realiza mediante otro circuito que recibe el nombre de **decodificador**, el cual se verá en el Capítulo 6. Un decodificador convierte la información codificada, como puede ser un número binario, en otra información no codificada, como lo es un número decimal. Por ejemplo, un tipo particular de decodificador convierte un código binario de 4 bits en su correspondiente dígito decimal.

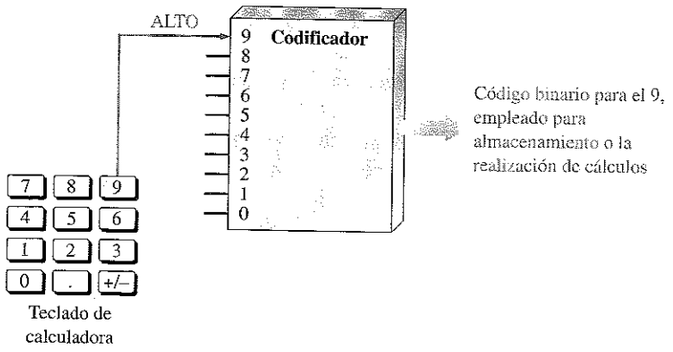


FIGURA 1.21
Un codificador utilizado para codificar una pulsación de una tecla de la calculadora en un código binario que se almacenará o se empleará en los cálculos.

La Figura 1.22 es una sencilla ilustración de un tipo de decodificador que se utiliza para activar un display de 7 segmentos. Cada uno de los siete segmentos del display está conectado a una línea de salida del decodificador. Cuando en las entradas del decodificador se presenta un determinado código binario, se activan las correspondientes líneas de salida y se iluminan los correspondientes segmentos del display para mostrar el dígito decimal que corresponde al código binario.

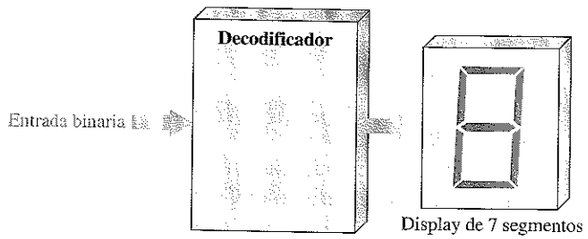


FIGURA 1.22
Decodificador que convierte un determinado código binario en un código 7-segmentos que permite visualizar el número decimal correspondiente.

Función de selección de datos

Existen dos tipos de circuitos dedicados a la selección de datos: el multiplexor y el demultiplexor. El **multiplexor** es un circuito lógico que pasa los datos digitales procedentes de varias líneas de entrada a una única línea de salida según una secuencia de tiempos específica. Funcionalmente, un multiplexor se puede representar mediante una operación de conmutación electrónica que secuencialmente conecta cada una de las líneas de entrada a la línea de salida. Un **demultiplexor** es un circuito que pasa los datos digitales procedentes de una línea de entrada a varias líneas de salida según una

determinada secuencia de tiempo. Esencialmente, el demultiplexor es un multiplexor invertido.

La multiplexación y demultiplexación se emplean cuando datos procedentes de distintas fuentes se tienen que transmitir a través de una línea hasta una localización distante, y deben redistribuirse a varios destinos. La Figura 1.23 ilustra este tipo de aplicación en la que los datos digitales de tres computadoras se envían a lo largo de una única línea hasta otras tres computadoras en otra localización.

En la Figura 1.23, los datos binarios de la computadora A se conectan a la línea de salida durante el intervalo de tiempo Δt_1, y se transmiten al demultiplexor conectado a la computadora D. Luego, durante el intervalo de tiempo Δt_2, el multiplexor conmuta a la entrada de la computadora B, mientras que el demultiplexor conmuta la salida a la computadora E. Durante el intervalo Δt_3, el multiplexor conmuta a la entrada de la computadora C y el demultiplexor conmuta la salida hacia la computadora F.

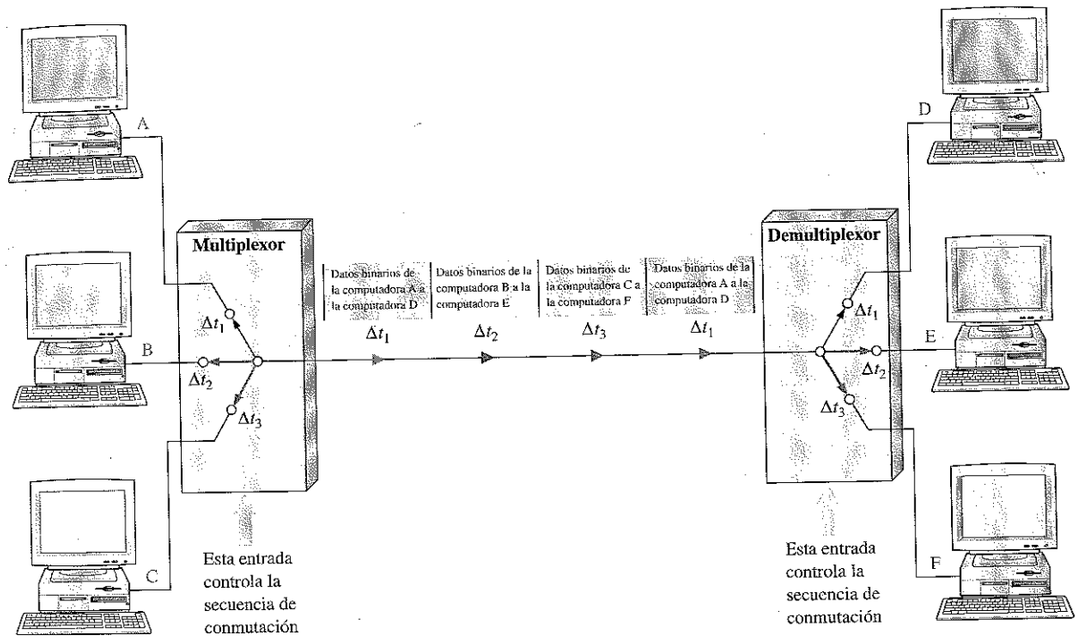


FIGURA 1.23
Ilustración de una aplicación básica de multiplexación/demultiplexación.

Resumiendo, durante el primer intervalo de tiempo, las dos primeras computadoras se comunican, y los datos pasan de la computadora A a la D. Durante el segundo intervalo de tiempo, las dos siguientes computadoras se comunican y los datos fluyen de B a E. Durante el tercer intervalo, las dos últimas computadoras se comunican, y los datos pasan de la computadora C a la F. A continuación, de nuevo se establece la comunicación entre las dos primeras computadoras y la secuencia se repite. Puesto que el tiempo se divide entre varios conjuntos de sistemas, en donde cada uno tiene un

NOTAS INFORMÁTICAS
Las memorias internas de la computadora, RAM y ROM, así como las memorias caché más pequeñas son memorias semiconductoras. Los registros de un microprocesador se construyen con dispositivos flip-flop semiconductores. En las unidades internas de disco duro, las unidades de disquete y de CD-ROM se utilizan memorias de disco magnéticas. Frecuentemente, también se utilizan las memorias de cinta magnética para la realización de copias de seguridad de los datos.

turno para enviar y recibir datos, el proceso se denomina *multiplexación por división en el tiempo (Time Division Multiplexing, TDM).*

Función de almacenamiento

La mayoría de los sistemas digitales requieren la función de **almacenamiento**, cuyo objetivo es mantener los datos binarios durante un periodo de tiempo. Algunos dispositivos de almacenamiento se usan para almacenamiento temporal, y otros para almacenamiento permanente. Un dispositivo de almacenamiento puede "memorizar" un bit o un grupo de bits y mantiene la información mientras sea necesario. Algunos tipos comunes de dispositivos de almacenamiento son: flip-flops, registros, memorias semiconductoras, discos magnéticos, cintas magnéticas y discos ópticos.

Flip-flops. El **flip-flop** es un circuito lógico biestable (dos estados estables), que puede almacenar un único bit en un determinado instante, bien un 1 o un 0. La salida de un flip-flop biestable indica qué bit está almacenado. Una salida a nivel ALTO indica que se ha almacenado un 1 y una salida a nivel BAJO indica que se ha almacenado un 0. Los flip-flops se implementan con puertas lógicas, y los veremos en el Capítulo 8.

Registros. Un **registro** se forma combinando varios flip-flop de manera que se puedan almacenar grupos de bits. Por ejemplo, un registro de 8 bits se construye a partir de ocho flip-flop. Además de almacenar bits, los registros pueden emplearse para desplazar los bits de una posición a otra dentro del propio registro o fuera del mismo a otro circuito; por tanto, estos dispositivos reciben el nombre de *registros de desplazamiento*, y se verán en el Capítulo 10.

Los dos tipos básicos de registros de desplazamiento son serie y paralelo. Como se muestra en la Figura 1.24, los bits se almacenan en un registro de desplazamiento serie uno a uno. Una buena analogía serían los pasajeros que entran en un autobús formando una sola fila frente a la puerta, y también bajarían del autobús formando una única fila.

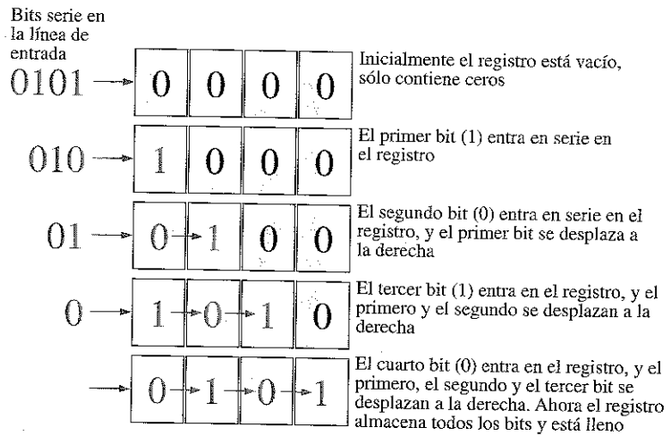


FIGURA 1.24
Ejemplo de funcionamiento de un registro de desplazamiento en serie de 4 bits. Cada bloque representa una "celda" de almacenamiento o flip-flop.

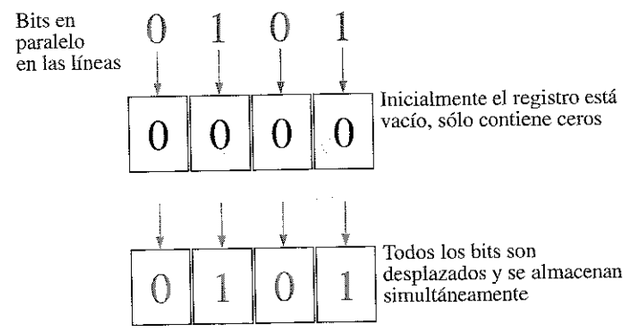


FIGURA 1.25
Ejemplo de funcionamiento de un registro de desplazamiento paralelo de 4 bits.

En un registro paralelo los bits se almacenan simultáneamente a partir de líneas paralelas, como se muestra en la Figura 1.25. En este caso, una buena analogía serían los pasajeros que se montan en una montaña rusa, montándose en los coches en paralelo.

Memorias semiconductoras. Las memorias semiconductoras son dispositivos típicamente utilizados para almacenar grandes cantidades de bits. En un tipo especial de memoria, denominada *memoria de sólo lectura* o ROM (Read Only Memory), los datos binarios se almacenan de forma permanente o semipermanente y no se pueden cambiar instantáneamente. En las memorias de acceso aleatorio o RAM (Random Access Memory), los datos binarios se almacenan temporalmente y pueden ser fácilmente modificados. Las memorias se tratarán en el Capítulo 12.

Memorias magnéticas. Las memorias de disco magnético se emplean para almacenamiento masivo de datos binarios. Ejemplos de estos dispositivos serían los disquetes o el disco duro de las computadoras. Los discos magneto-ópticos utilizan haces láser para almacenar y recuperar datos. Las cintas magnéticas todavía se emplean en aplicaciones de memorias y para la realización de copias de seguridad de datos procedentes de otros dispositivos de almacenamiento.

Función de recuento

La función de contar (recuento) es muy importante en los sistemas digitales. Existen muchos tipos de **contadores** digitales, pero su objetivo básico es contar sucesos representados por cambios de nivel o impulsos, o generar una secuencia de códigos particular. Para contar, el contador debe "recordar" el número actual, con el fin de poder pasar correctamente al siguiente número de la secuencia. Por tanto, la capacidad de almacenamiento es una característica importante de todos los contadores y, generalmente, se utilizan los flip-flop para su implementación. La Figura 1.26 ilustra la idea básica del funcionamiento del contador. En el Capítulo 9 se abordarán los contadores.

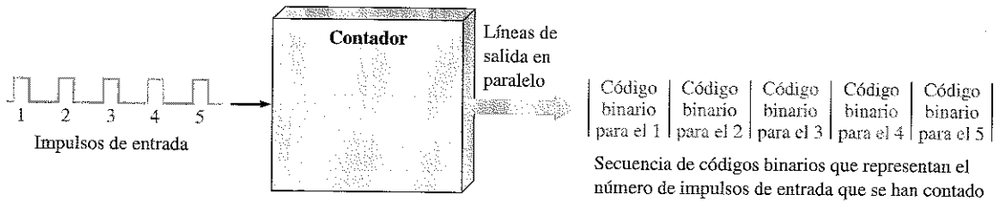


FIGURA 1.26
Funcionamiento básico del contador.

REVISIÓN DE LA SECCIÓN 1.4

1. ¿Qué hace un comparador?
2. ¿Cuáles son las cuatro operaciones aritméticas básicas?
3. Describir qué es la codificación y dar un ejemplo.
4. Describir qué es la decodificación y dar un ejemplo.
5. Explicar el objetivo básico de la multiplexación y la demultiplexación.
6. Nombrar cuatro tipos de dispositivos de almacenamiento.
7. ¿Qué hace un contador?

1.5 ■ CIRCUITOS DIGITALES INTEGRADOS

Todos los elementos y funciones lógicos que hemos tratado (y muchos más) están disponibles como circuitos integrados (CI). Los modernos sistemas digitales utilizan CI casi exclusivamente en su diseño debido a su reducido tamaño, alta fiabilidad, bajo coste y bajo consumo de potencia. Es importante ser capaz de reconocer los encapsulados de los CI y saber cómo se numeran sus pines, así como familiarizarse con la forma en que la complejidad de los circuitos y su tecnología determinan las distintas clasificaciones de circuitos integrados.

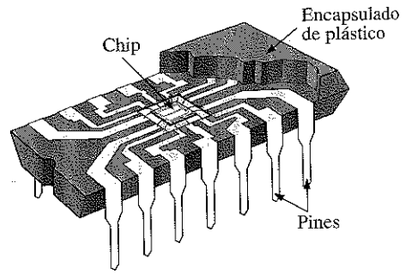
Al finalizar esta sección, el lector deberá ser capaz de:

■ Reconocer las diferencias entre dispositivos de inserción y de montaje superficial. ■ Identificar los encapsulados DIP (*Dual In-line Package*). ■ Identificar los encapsulados SOIC (*Small-Outline IC*). ■ Identificar los encapsulados PLCC (*Plastic Leaded Chip Carrier*). ■ Identificar los encapsulados LCCC (*Less Ceramic Chip Carrier*). ■ Identificar los encapsulados flat-pack (FP). ■ Determinar la numeración de los pines en los distintos tipos de encapsulados de los CI. ■ Explicar la clasificación de los circuitos integrados según su complejidad.

Un **circuito integrado (CI)** monolítico es un circuito electrónico construido enteramente sobre un pequeño chip de silicio. Todos los componentes que conforman el circuito: transistores, diodos, resistencias y condensadores, son parte integrante de un único chip.

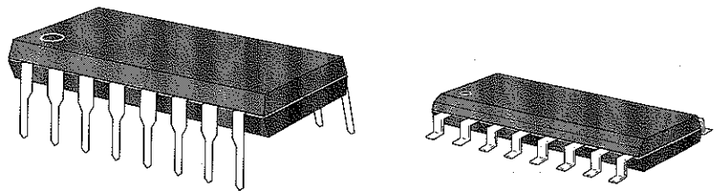
La Figura 1.27 muestra una sección de un tipo de encapsulado de CI, donde se ve el chip del circuito dentro del encapsulado. Los terminales del chip se conectan a los pines del encapsulado para permitir las conexiones de entrada y salida al mundo exterior.

FIGURA 1.27
Sección de un encapsulado de CI que presenta el chip montado en el interior, con conexiones a los pines de entrada y salida.



Encapsulados de los circuitos integrados

Los encapsulados de los circuitos integrados se clasifican según la forma en que se montan sobre las tarjetas de circuito impreso (PCB, *printed circuit board*) y pueden ser de inserción o de montaje superficial. Los componentes de inserción tienen unos pines (patas) que se insertan en los taladros de la tarjeta de circuito impreso y se sueldan a las pistas por la cara opuesta. El encapsulado de inserción más típico es el encapsulado **DIP** (*dual in-line package*), que se muestra en la Figura 1.28(a).



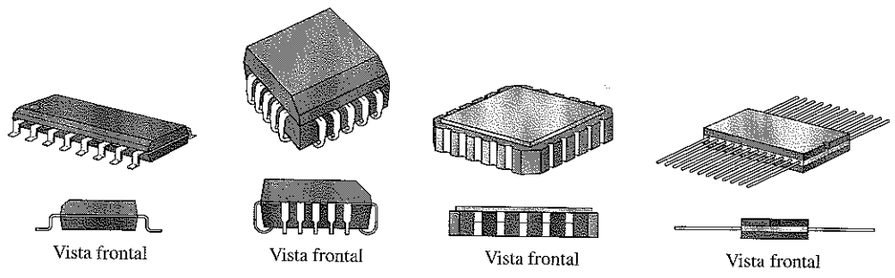
(a) Encapsulado DIP (b) Encapsulado SOIC

FIGURA 1.28
Ejemplos de dispositivos de inserción y de montaje superficial. El encapsulado DIP es más grande que el SOIC y tienen el mismo número de patas. Este DIP en concreto mide aproximadamente 2 cm de largo, y el SOIC mide alrededor de 1 cm.

Otra técnica de encapsulado de CI es la **tecnología de montaje superficial (SMT, Surface-Mount Technology)**. El montaje superficial es un método más moderno, que permite ahorrar espacio, alternativo al montaje de inserción. Los taladros de las tarjetas de circuito impreso son innecesarios en SMT. Los pines de los encapsulados de montaje superficial se sueldan directamente a las pistas de una cara de la tarjeta, dejando la otra cara libre para añadir circuitos adicionales. Además, para un circuito con el mismo número de pines, un encapsulado de montaje superficial es mucho más pequeño que un encapsulado DIP, ya que los pines se sitúan mucho más cercanos entre sí. Un ejemplo de encapsulado de montaje superficial son los SOIC, mostrados en la Figura 1.28(b).

Cuatro tipos comunes de encapsulados SMT son: el **SOIC** (small-outline IC), el **PLCC** (plastic leaded chip carrier), el **LCCC** (leadless ceramic chip carrier) y los encapsulados **flat-pack** (FP). Estos tipos de encapsulado están disponibles en distintos tamaños dependiendo del número de pines (cuanto más complicado sea el circui-

to, mayor número de pines se requiere). Ejemplos de cada tipo se muestran en la Figura 1.29. Como puede verse, los pines del SOIC tienen forma de "alas de gaviota". Los terminales de los PLCC envuelven la parte inferior del encapsulado en forma de J. Por el contrario, los terminales del LCCC tienen contactos metálicos que se introducen en el cuerpo cerámico. Los terminales de un flat-pack se extienden en línea recta hacia el exterior del encapsulado. Otras variantes de los encapsulados SMT son: el encapsulado **SSOP** (shrink small-outline package), el encapsulado **TSSOP** (thin shrink small-outline package) y el encapsulado **TVSOP** (thin very small-outline package).

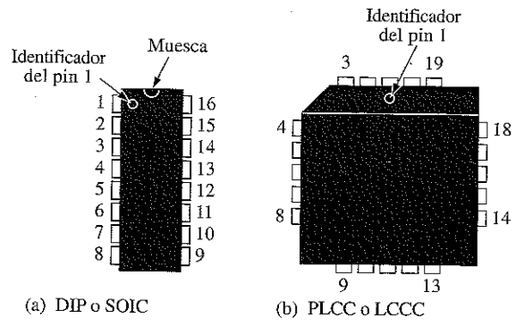


(a) SOIC con terminales en forma de "alas de gaviota". (b) PLCC con terminales en forma de J. (c) LCCC sin terminales (los contactos son parte de la caja). (d) Flat pack (FP) con terminales rectos.

FIGURA 1.29
Ejemplos de configuraciones de encapsulados de montaje superficial.

Numeración de los pines

Todos los encapsulados de CI tienen un formato estándar para numerar los pines (terminales). En la Figura 1.30(a) se muestra el encapsulado de 16 pines para los encapsulados DIP, SOIC y flat pack, indicándose la numeración de sus pines. En la parte superior del encapsulado, se designa el pin 1 mediante un identificador que puede ser un pequeño punto, una muesca o una esquina biselada. El punto siempre se encuentra junto al pin 1. También, si la muesca se coloca hacia arriba, el pin 1 es siempre el pin superior de la izquierda, tal como se indica. Empezando por el pin 1, el número de pin aumenta a medida que se desciende, y se continúa por el lado opuesto en sentido ascendente. El número mayor de pin es siempre el que se sitúa a la derecha de la muesca o el que está enfrente del punto.



(a) DIP o SOIC (b) PLCC o LCCC

FIGURA 1.30
Numeración de los pines para dos tipos de encapsulados estándar de CI. Se muestran las vistas superiores.

Los encapsulados PLCC y LCCC tienen terminales en sus cuatro costados. El pin 1 se identifica mediante un punto u otra marca y se encuentra situado en el centro de uno cualquiera de los lados del chip. La numeración de los terminales se incrementa en sentido contrario a las agujas del reloj mirando la parte superior del encapsulado. El pin de mayor numeración está siempre a la derecha del pin 1. La Figura 1.30(b) ilustra este formato para un encapsulado PLCC de 20 pines.

Clasificación de los circuitos integrados según su complejidad

Los circuitos integrados se clasifican de acuerdo a su complejidad. Aquí se enumeran de menor a mayor complejidad. La clasificación por complejidad establecida aquí para SSI, MSI, LSI, VLSI y ULSI está generalmente aceptada, aunque las definiciones pueden variar de una fuente a otra.

- La **integración a baja escala (Small-Scale Integration, SSI)** describe los circuitos integrados que contienen hasta 12 puertas equivalentes en un único chip, e incluyen puertas básicas y flip-flops.
- La **integración a media escala (Medium-Scale Integration, MSI)** describe los circuitos integrados que poseen de 12 a 99 puertas equivalentes en un único chip. Incluyen funciones lógicas, tales como codificadores, decodificadores, contadores, registros, multiplexores, circuitos aritméticos, pequeñas memorias y otros.
- La **integración a gran escala (Large-Scale Integration, LSI)** es una categoría que incluye CI que contienen entre 100 y 9.999 puertas equivalentes por chip, incluyendo memorias.
- La **integración a muy gran escala (Very Large-Scale Integration, VLSI)** incluye circuitos integrados con un número de puertas equivalentes desde 10.000 hasta 99.999 por chip.
- La **integración a ultra escala (Ultra Large-Scale Integration, ULSI)** nos describe memorias de gran capacidad, grandes **microprocesadores** y potentes computadoras en un único chip. Esta clasificación designa CI que contienen más de 100.000 puertas equivalentes por chip.

Tecnologías de circuitos integrados

Los tipos de transistores con los que se implementan los circuitos integrados pueden ser transistores bipolares o MOSFET (Metal-Oxide Semiconductor Field-Effect Transistor, transistores de efecto campo por unión metal-óxido-semiconductor). Dos tecnologías de circuitos digitales que usan transistores de uniones bipolares son: TTL (Transistor-Transistor Logic, lógica transistor-transistor) y ECL (Emitter-Coupled Logic, lógica de emisor acoplado). De éstas, la TTL es la más ampliamente utilizada. La mayor parte de las tecnologías de circuitos que utilizan MOSFET son de tipo CMOS (Complementary MOS, MOS complementario) o NMOS (*n*-channel MOS, MOS canal-*n*). Los microprocesadores utilizan la tecnología CMOS.

Todas las puertas lógicas y otras funciones pueden implementarse con cualquier tipo de tecnología de circuitos. Generalmente, los circuitos SSI y MSI están disponibles en TTL y CMOS, y los circuitos LSI, VLSI y ULSI se implementan normalmente con CMOS o NMOS, ya que estas tecnologías requieren

una menor superficie de chip y consumen menos potencia. En el Capítulo 3 se tratan en más detalle estas tecnologías sobre circuitos. Además, el Capítulo 15 proporciona una introducción completa a estas tecnologías de circuitos integrados, y es un capítulo que puede consultarse en cualquier punto de la secuencia del resto de los capítulos.

Consejos Prácticos

Debido a su estructura particular, los dispositivos MOS son muy sensibles a las cargas estáticas y pueden resultar dañados por las descargas electrostáticas, si no se manipulan adecuadamente. Cuando trabaje con dispositivos MOS deberá tomar las siguientes precauciones:

- ❑ Los dispositivos MOS deben ser suministrados y almacenados en espuma conductiva.
- ❑ Todos los instrumentos y bancos metálicos utilizados en las pruebas deberían estar conectados a tierra.
- ❑ Las herramientas de trabajo deberían conectarse a tierra mediante un cable y varias resistencias en serie de alto valor.
- ❑ No retire un dispositivo MOS (o cualquier dispositivo) de un circuito mientras que la alimentación continua esté conectada.
- ❑ No conecte tensiones de señal o de corriente alterna a un dispositivo MOS cuando la alimentación continua esté apagada.

REVISIÓN DE LA SECCIÓN 1.5

1. ¿Qué es un circuito integrado?
2. Definir los términos DIP, SMT, SOIC, SSI, MSI, LSI, VLSI y ULSI.
3. Generalmente, ¿a qué clasificación pertenece un CI con el siguiente número de puertas equivalentes?
   (a) 75 **(b)** 500 **(c)** 10 **(d)** 15.000 **(e)** 200.000

1.6 ■ INSTRUMENTACIÓN PARA PRUEBAS Y LOCALIZACIÓN DE AVERÍAS

La localización de averías es la técnica de aislar, identificar y corregir sistemáticamente un fallo en un circuito o sistema. Existe una gran variedad de instrumentos que se pueden utilizar en la localización de averías y pruebas de sistemas digitales. En esta sección, se presentan algunos equipos típicos.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Reconocer los controles más comunes de un osciloscopio. ■ Distinguir entre un osciloscopio analógico y un osciloscopio digital. ■ Determinar la amplitud, periodo, frecuencia y ciclo de trabajo de una señal digital con un osciloscopio. ■ Saber qué función tiene un analizador lógico. ■ Enumerar los formatos más comunes de un analizador lógico. ■ Explicar cómo se usan la sonda lógica, el pulsador lógico y la sonda de corriente. ■ Conocer las funciones de una fuente de alimentación de continua, un generador de funciones y un multímetro digital.

El osciloscopio

El osciloscopio es uno de los instrumentos más ampliamente utilizados con propósitos de prueba y **localización de averías**. Básicamente, es un dispositivo con pantalla gráfica que traza una gráfica de una señal eléctrica en su pantalla. En la mayor parte de las aplicaciones, las gráficas se muestran como señales en función del tiempo. El eje vertical de la pantalla representa la tensión y el eje horizontal representa el tiempo. En las aplicaciones digitales, se pueden visualizar las formas de onda de pulsos y se pueden medir parámetros como la amplitud, el periodo, la frecuencia, los tiempos de subida y bajada y el ciclo de trabajo. También se pueden visualizar simultáneamente dos formas de onda de pulsos (con osciloscopios de dos canales), con el fin de determinar su relación en función del tiempo.

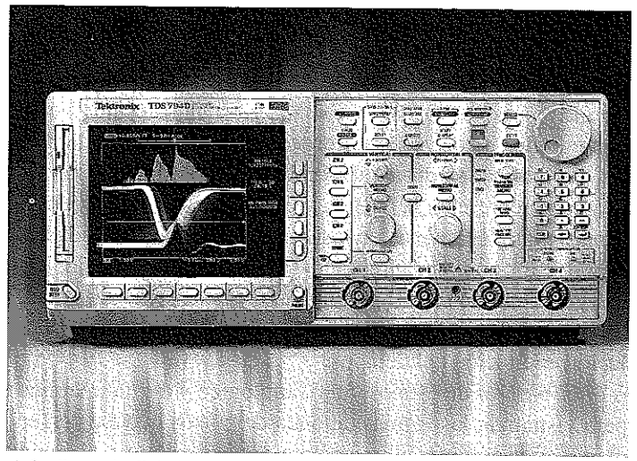
Existen dos tipos básicos de osciloscopio: analógico y digital. Ambos tipos se pueden utilizar para visualizar las formas de onda de pulsos que existen en los circuitos digitales. Un osciloscopio analógico funciona aplicándole la señal de tensión que se va a medir a un haz de electrones que oscilará a lo largo de la pantalla. La tensión medida hace que el haz se deflecte arriba y abajo de forma proporcional a la cantidad de tensión, trazando inmediatamente la forma de onda en la pantalla. Un osciloscopio digital muestrea la tensión de la señal y utiliza un conversor analógico-digital (ADC) para convertir la tensión que se va a medir en información digital. Esta información se utiliza a continuación para reconstruir la forma de onda en la pantalla. En la Figura 1.31 se muestran osciloscopios de sobremesa y portátiles típicos.

Aunque todavía se fabrican osciloscopios analógicos y muchos de ellos todavía se utilizan, el osciloscopio digital se utiliza cada vez más y está empezando a reemplazar a los antiguos instrumentos analógicos.

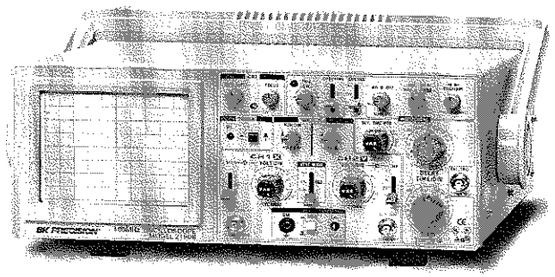
El osciloscopio analógico

La Figura 1.32 muestra una vista genérica del panel frontal de un osciloscopio analógico típico. Dispone de dos controles que indican los voltios por división (VOLTS/DIV), para cada uno de los dos canales. Estos controles determinan la atenuación o amplificación de la señal y cada uno de los botones de dial indica el número de voltios (V) o milivoltios (mV) representado por cada división vertical de la pantalla. Normalmente, existen dos escalas en el botón VOLTS/DIV, uno para la sonda de tensión x10 y otro para la sonda de tensión x1. La sonda más comúnmente utilizada es la sonda x10, que atenúa la señal de entrada por diez, para reducir los efectos de carga sobre el circuito. Cuando se utiliza la sonda x10, se debe leer la medida de los voltios por división en la escala x10.

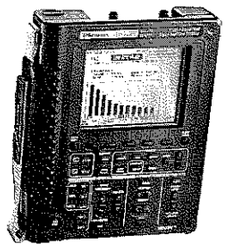
La base de tiempos o control SEC/DIV establece el número de segundos (s), milisegundos (ms) o microsegundos (μs) que representa cada división horizontal según indica la escala del dial. Se puede medir el periodo de una forma de onda de impulsos en la escala horizontal y, a continuación, calcular la frecuencia.



a) Osciloscopio digital



b) Osciloscopio analógico



c) Osciloscopio portátil

FIGURA 1.31
Osciloscopios típicos. Los modelos (a) y (c) son propiedad de Tektronix, Inc. 1998. Todos los derechos reservados. El modelo (b) es cortesía de B&K Precision Corp.

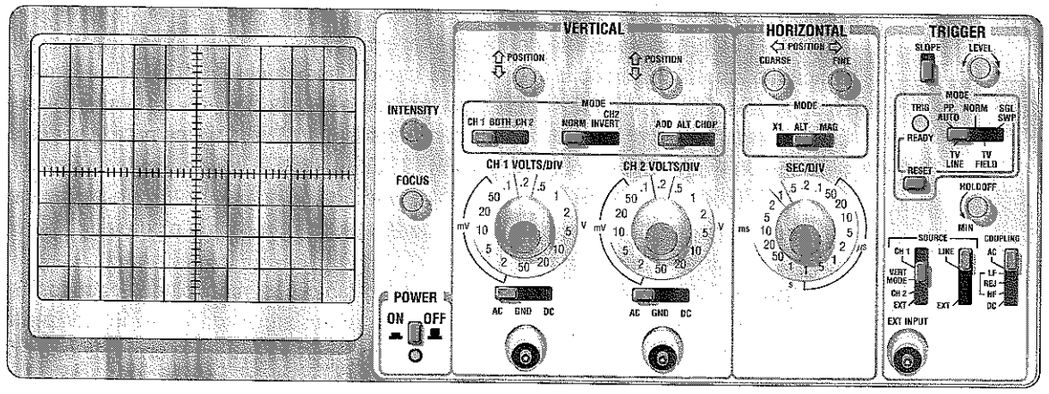


FIGURA 1.32
Típico osciloscopio analógico de doble canal.

EJEMPLO 1.3 En función de las escalas indicadas en el dial de control, determinar la amplitud y el periodo del tren de impulsos mostrado en la pantalla del osciloscopio analógico, que se muestra en la Figura 1.33. Calcular también la frecuencia. Se supone que las medidas se han realizado con una sonda de tensión x10.

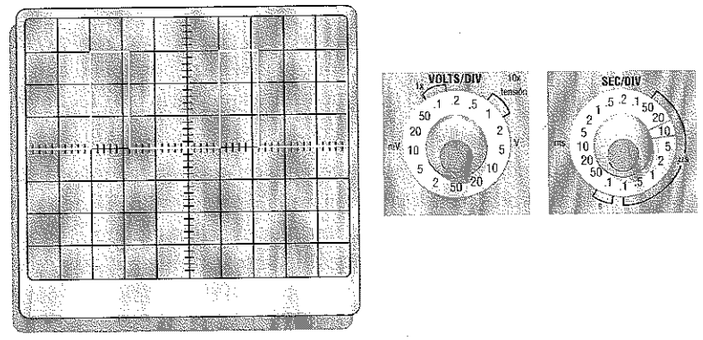


FIGURA 1.33

Solución. El control VOLTS/DIV se encuentra en la escala de 1V. Los pulsos tienen una altura de tres divisiones. Dado que cada división representa 1V, la amplitud del impulso es

$$\text{Amplitud} = (3 \text{ divisiones})\ (1 \text{ voltio/división}) = \mathbf{3\ V}$$

La escala de control SEC/DIV está en 10 µs. Un ciclo completo de la forma de onda (desde el inicio del primer impulso al inicio del siguiente) cubre cuatro divisiones; por tanto, el periodo es

$$\text{Periodo} = (4 \text{ divisiones})\ (10\ \mu\text{s/división}) = \mathbf{40\ \mu s}$$

La frecuencia se calcula del siguiente modo:

$$f = \frac{1}{T} = \frac{1}{40\ \mu s} = \mathbf{25\,kHz}$$

Problema relacionado. Si la escala del control VOLTS/DIV marca 4 V y el control SEC/DIV marca 2 ms, determinar la amplitud y el periodo del pulso mostrado en la pantalla de la Figura 1.33. Suponemos que se utiliza la sonda x10.

Otros controles básicos. Las siguientes descripciones hacen referencia al osciloscopio analógico de la Figura 1.32. Sin embargo, muchos de estos controles también son habituales en los osciloscopios digitales, aunque algunas de las funciones son accesibles a través de un sistema de menús.

Interruptor de encendido (Power switch). Este interruptor enciende o apaga el osciloscopio. Una luz indica si está encendido.

Intensidad (Intensity). El botón de control de la intensidad varía el brillo de la traza en la pantalla. Debe utilizarse con precaución ya que la intensidad no debe ser demasiado alta durante un periodo de tiempo muy largo, especialmente si el haz forma un punto inmóvil sobre la pantalla.

Enfoque (Focus). Este control enfoca el haz de forma que converja a un punto fino en la pantalla. Una condición de desenfoque origina una imagen borrosa.

Posiciones horizontal y vertical (Horizontal and vertical positions). Estos controles se emplean para desplazar una traza horizontal a lo largo de la pantalla y hacia arriba y hacia abajo para medirla y apreciarla más fácilmente.

Conmutador AC-GND-DC. Este selector permite un acoplamiento de la señal de entrada en continua, alterna o de masa. El acoplamiento AC elimina cualquier componente continua de la señal de entrada. El acoplamiento DC permite también visualizar valores de continua, y la posición GND establece como referencia una señal de 0 V en la pantalla. *Para señales digitales debería emplearse el acoplamiento en continua.*

Entradas de señal (Signal inputs). Las tensiones que se visualizan se conectan al canal 1 (CH1) y/o al canal 2 (CH2). Como hemos mencionado anteriormente, estas conexiones de entrada se realizan mediante una sonda atenuadora que minimiza los efectos de carga del osciloscopio sobre el circuito que se está midiendo. El tipo común de sonda de tensión de osciloscopio atenúa (reduce) la tensión de la señal por un factor de diez y se denomina sonda x10. Las sondas x1 también se utilizan, ocasionalmente, para la medida de tensiones muy pequeñas. En la Figura 1.34 se muestra una sonda de tensión típica.

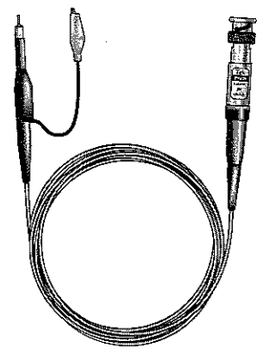


FIGURA 1.34
Sonda de tensión de osciloscopio.

Selectores de modo (mode switches). Estos interruptores permiten visualizar uno o ambos canales de las señales de entrada, invertir la señal del canal dos (INVERT), sumar dos formas de onda (ADD) y seleccionar entre los modos de barrido alternado (ALT) y troceado (CHOP). *Como regla general, cuando la frecuencia es superior a 100 Hz, se selecciona el modo ALT; en caso contrario, se utiliza el modo CHOP.*

Controles de disparo (trigger controls). Estos controles permiten que el osciloscopio sea disparado desde distintas fuentes. El disparo hace que la traza inicie su barrido a lo largo de la pantalla. El disparo se puede realizar mediante una señal interna, una señal externa o a partir de la tensión de línea. Los modos típicos de disparo son *normal, automático, barrido único* y *TV*. En el modo automático, se produce una oscilación en ausencia de una señal de disparo adecuada. En el modo normal, debe existir una señal de disparo para que se produzca la oscilación. El modo TV proporciona el disparo sobre el campo TV con las señales de línea TV. El interruptor de pendiente (*slope*) establece que el disparo se produzca sobre la pendiente positiva o negativa de la forma de onda de disparo. El control de nivel (*level*) selecciona el nivel de tensión de la señal de disparo en la que se producirá el disparo. Básicamente, los controles de disparo sirven para controlar la sincronización de la forma de onda horizontal del osciloscopio y las señales de entrada. Como resultado, se muestra una señal de entrada estable en la pantalla, que no tiene deriva.

Osciloscopio digital

En la Figura 1.35 se muestra una vista genérica del panel frontal de un osciloscopio digital típico. Observe que las secciones vertical y horizontal constan de un control de posición, un botón de menú y un control de escala.

En la sección *vertical*, los controles se aplican a cualquiera de los canales en función de los botones de selección de canal. El control de posición permite desplazar la forma de onda hacia arriba y hacia abajo por la pantalla. El botón de menú permite seleccionar diversos elementos que aparecen en la pantalla, tal como el acoplamiento (en alterna, continua o tierra), la inversión de la forma de onda y otros parámetros. El control de escala ajusta el número de voltios representado por cada división vertical, aunque en este tipo de osciloscopio los voltios por división para un canal se ven mejor en pantalla que leyendo en el dial. Las sondas para osciloscopios digitales tienen un elemento especial que detecta si la sonda está en escala x1 o x10, ajustando automáticamente las lecturas.

En la sección *horizontal*, los controles se aplican a ambos canales. El control de posición permite desplazar la forma de onda horizontalmente por la pantalla. El botón de menú proporciona una selección de diversos elementos que se muestran en pantalla, tal como el barrido principal o retardado y otros parámetros. El control de escala ajusta el tiempo representado por cada división horizontal, aunque en este tipo de osciloscopio se visualizan mejor las divisiones por segundo en la pantalla que en el dial.

En la sección *trigger* (disparo), el control de nivel (*level*) determina dónde se produce el punto de disparo. El botón de menú proporciona una selección de elementos que se muestran en pantalla, tal como el modo normal, modo automático, modo vídeo, origen de disparo, pendiente de disparo y otros parámetros.

Muchos osciloscopios digitales disponen de una unidad de disquete que permite almacenar las formas de onda para visualizarlas, analizarlas o importarlas, posteriormente, a un programa de composición u otros programas ubicados en una computadora.

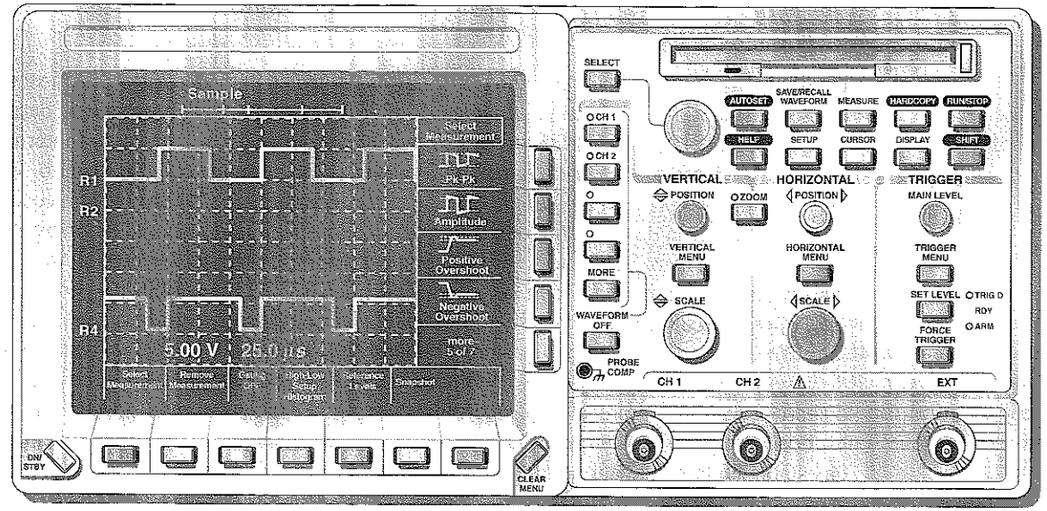
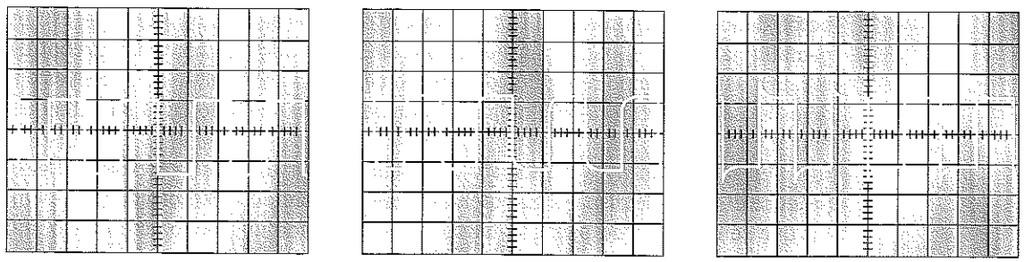


FIGURA 1.35
Típico osciloscopio digital de dos canales.

Consejos Prácticos

Algunas de las medidas que se realizan con un osciloscopio requerirán una sonda atenuadora x10, que reduzca la tensión de la señal y la carga del circuito por un factor de 10. Antes de llevar a cabo ninguna medida, debería asegurarse de que la sonda está compensada, ya que una sonda incorrectamente compensada puede distorsionar las formas de onda que se muestran en la pantalla del osciloscopio. En la mayoría de las sondas existe un ajuste mediante tornillo u otro medio para propósitos de compensación y que normalmente está calibrado para obtener una señal cuadrada en la pantalla del osciloscopio. Las formas de onda del osciloscopio que se presentan a continuación muestran tres posibles condiciones: correctamente compensada, subcompensada y sobrecompensada.



Correctamente compensada | Subcompensada | Sobrecompensada

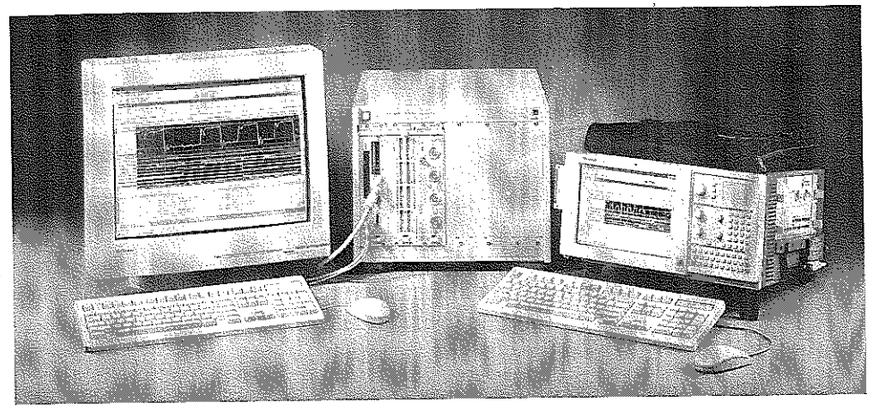


FIGURA 1.36
Analizadores lógicos típicos. Propiedad de Tektronix, Inc. Todos los derechos reservados. Reproducido con su permiso.

El analizador lógico

En la Figura 1.36 se muestran varios analizadores lógicos. Este instrumento puede detectar y presentar datos digitales en distintos formatos.

Formato de osciloscopio. El analizador lógico puede utilizarse para presentar en pantalla uno o dos tipos de señales, de forma que las características de los impulsos o de las señales se puedan medir.

Formato de cronograma. El analizador lógico puede mostrar típicamente hasta 16 señales en su adecuada relación de tiempos, de forma que se pueden analizar conjuntos de señales y determinar cómo varían en el tiempo unas respecto de otras.

Formato de estados. Un analizador lógico puede presentar datos binarios en forma tabular. Por ejemplo, pueden examinarse varias posiciones de memoria de un sistema basado en microprocesador para determinar su contenido. Los datos pueden presentarse en diversos sistemas de numeración y códigos tales como binario, hexadecimal, octal, código decimal binario (BCD) y ASCII. Estos sistemas de numeración y códigos serán los temas que trataremos en el siguiente capítulo.

Sonda lógica, pulsador lógico y sonda de corriente

La sonda lógica es una herramienta conveniente y manejable que proporciona un medio para la localización de averías en un circuito digital, detectando las condiciones en un punto del circuito, como ilustra la Figura 1.37. La sonda puede detectar niveles de tensión altos, bajos, impulsos aislados, impulsos repetitivos y circuitos abiertos en una tarjeta de circuito impreso. La luz de la sonda indica la condición en que se encuentra un determinado punto del circuito, tal y como se indica en la figura. El pulsador lógico genera trenes de impulsos repetitivos que pueden aplicarse a cualquier punto de un circuito. Se pueden aplicar impulsos en un punto del circuito con el pulsador y comprobar el resultado en algún otro punto con una sonda lógica. El pulsador lógico se puede usar también junto con la sonda de corriente, como se indica en la

Figura 1.38. La sonda de corriente detecta si existe en la línea corriente pulsatoria, y es particularmente útil para localizar cortocircuitos en una tarjeta de circuito impreso.

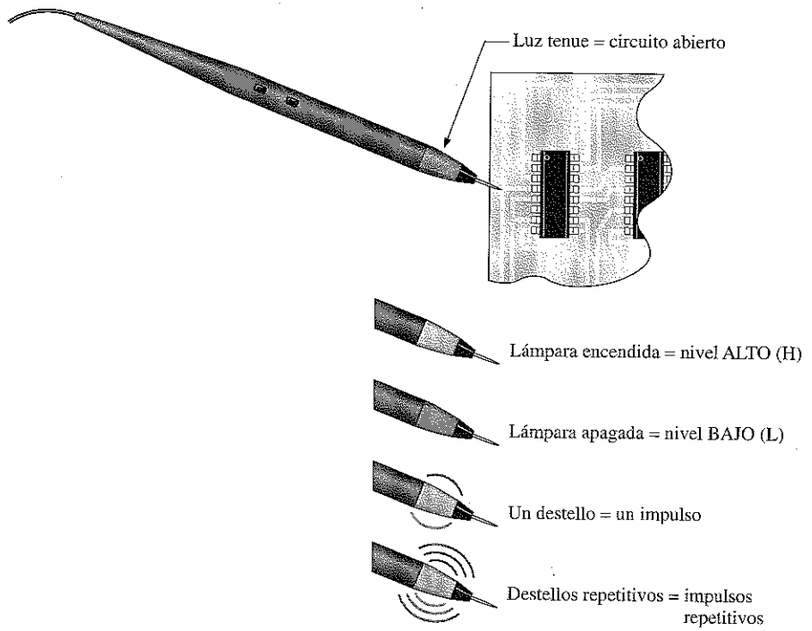


FIGURA 1.37
Utilización de una sonda lógica para detectar distintas condiciones de tensión en un determinado punto del circuito.

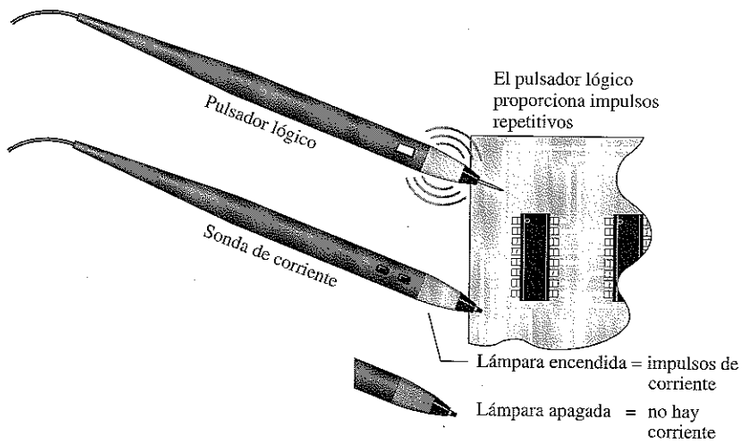


FIGURA 1.38
Utilización de un pulsador lógico y una sonda de corriente para generar impulsos en un punto determinado y comprobar la corriente resultante en otra parte del circuito.

Fuente de alimentación DC

La fuente de alimentación DC es un instrumento indispensable en cualquier banco de pruebas. La fuente de alimentación convierte la tensión alterna que obtenemos de cualquier enchufe de pared en una tensión continua regulada. Todos los circuitos digitales requieren tensión continua para funcionar. Por ejemplo, los circuitos TTL y muchos circuitos CMOS necesitan aproximadamente +5 V. La fuente de alimentación se utiliza cuando se trabaja con un nuevo prototipo o cuando se extrae una tarjeta de circuito impreso de algún sistema para probarlo, y ya no se está alimentando con la fuente de alimentación interna del sistema. En la Figura 1.39 se muestra una típica fuente de alimentación de un banco de pruebas.

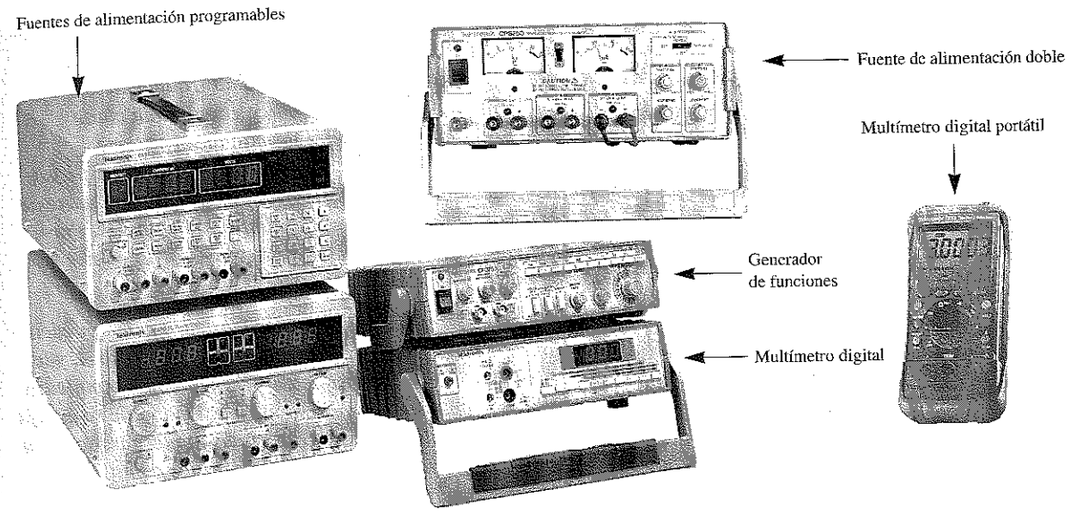


FIGURA 1.39
Instrumentos de prueba típicos. Copyright 1998 Tektronix, Inc. Todos los derechos reservados. Reproducida con su permiso.

Generador de funciones

El generador de funciones es una fuente de señales muy versátil que proporciona trenes de impulsos, así como señales sinusoidales y triangulares. Muchos generadores de funciones poseen salidas de lógica compatible para proporcionar señales de niveles adecuados a las entradas de los circuitos digitales, y así poder probar su funcionamiento. En la Figura 1.39 se muestra un tipo de generador de funciones.

Multímetro digital

Ningún banco de pruebas está completo sin un multímetro digital (DMM, digital multimeter). Este instrumento se emplea para medir tensiones alterna y continua, corrien-

tes alterna y continua, y resistencias. La Figura 1.39 muestra un multímetro de un banco de trabajo y un multímetro portátil.

REVISIÓN DE LA SECCIÓN 1.6

1. ¿Cuántas divisiones principales horizontales hay en la pantalla de un osciloscopio?
2. ¿Cuántas divisiones principales verticales hay en la pantalla de un osciloscopio?
3. ¿Qué tipo de instrumento puede mostrar hasta 16 tipos de formas de onda en su pantalla?
4. ¿Qué indica que la luz de una sonda lógica esté parpadeando repetitivamente?

1.7 ■ APLICACIÓN A LOS SISTEMAS DIGITALES

En esta sección, se presenta una interesante aunque simplificada aplicación de un sistema que contiene las funciones y elementos lógicos que se han tratado en las secciones anteriores. Es importante que un técnico electrónico comprenda cómo varias funciones digitales pueden operar conjuntamente formando un sistema completo para realizar una tarea específica. También es importante empezar a pensar en términos de funcionamiento a nivel de sistema ya que, en la práctica, gran parte de su trabajo va a implicar el tratar con sistemas y no con funciones individuales. Por supuesto, para comprender los sistemas, primero se deben entender las funciones y elementos básicos que los conforman.

Esta sección introduce el concepto de sistema. Este ejemplo le mostrará cómo pueden operar conjuntamente las funciones lógicas para realizar una tarea de alto nivel, y le permitirá comenzar a pensar a nivel de sistema. El sistema específico que se empleará aquí para ilustrar el concepto de sistema no es necesariamente el método que se usaría en la práctica, aunque podría serlo. En las aplicaciones industriales modernas, como la que se trata aquí, a menudo se emplean ciertos instrumentos conocidos como controladores programables.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Comenzar a pensar en términos de funcionamiento a nivel de sistema. ■ Explicar la finalidad de cada función lógica en el sistema de ejemplo. ■ Comprender cómo trabajan juntas las distintas funciones para llevar a cabo la tarea deseada.

El sistema

Imaginemos que una empresa farmacéutica utiliza el sistema descrito en la Figura 1.40 para contar y envasar automáticamente pastillas. Las pastillas se introducen en un gran embudo conductor. El estrecho cuello del embudo sólo permite que caiga una única pastilla dentro del bote que se encuentra sobre la cinta transportadora.

El sistema digital controla el número de pastillas que caen dentro de cada bote y muestra el total de pastillas actualizado continuamente en un display próximo a la línea de montaje, así como en una localización remota en otro lugar de la planta de fabricación. Como puede observarse, este sistema utiliza todas las funciones lógicas básicas que se han introducido en la Sección 1.4, y su único propósito es mostrar cómo se pueden combinar estas funciones para conseguir los resultados deseados.

El funcionamiento general es el siguiente: un sensor óptico en la parte inferior del embudo detecta cada pastilla que pasa y genera un impulso eléctrico. Este impulso llega al contador y hace que éste se incremente en una unidad, por lo que mientras que el bote se está llenando, el contador almacena la representación binaria del número de pastillas que hay en el bote. Este número binario se transfiere desde el contador a través de las líneas en paralelo hasta la entrada *B* del comparador. En la entrada *A* del comparador se aplica el número binario preseleccionado igual a la cantidad de pastillas que hay que introducir en cada bote. Este número preseleccionado proviene del teclado numérico y de su circuitería asociada, que incluye el codificador A, el registro A y el convertidor de código A. Cuando el número deseado de pastillas se introduce en el teclado, se codifica y se almacena en el registro paralelo A hasta que se requiere cambiar la cantidad de pastillas por bote.

Supongamos por ejemplo que cada bote va a contener 50 pastillas. Cuando el número del contador alcanza este valor, la salida $A = B$ del comparador se pone a nivel ALTO, indicando que el bote está lleno.

La salida a nivel ALTO del comparador hace que se cierre inmediatamente la válvula del cuello del embudo para detener el flujo de pastillas y, al mismo tiempo, activa la cinta transportadora para mover el siguiente bote y situarlo debajo del embudo. Cuando el siguiente bote está colocado correctamente debajo del cuello del embudo, el circuito de control de la cinta transportadora genera un impulso que pone a cero el contador. La salida $A = B$ del comparador pasa a nivel BAJO, abriendo la válvula del embudo para reiniciar el flujo de pastillas.

En la parte del sistema correspondiente al display, el número contenido en el contador se transmite en paralelo a la entrada *A* del sumador. En la entrada *B* del sumador se aplica el número total de pastillas envasadas hasta el último bote que se ha llenado, que procede del registro paralelo B. Por ejemplo, si se han llenado diez botes, y cada bote contiene 50 pastillas, el registro B contiene la representación binaria del número 500. Luego, cuando se ha llenado el siguiente bote, el número binario correspondiente a 50 se presenta en la entrada *A* del sumador, y en la entrada *B* se encuentra el número binario correspondiente a 500. El sumador genera una nueva suma, cuyo resultado es 550, que se almacena en el registro B, reemplazando a la suma anterior de 500.

El número binario contenido en el registro B se transfiere en paralelo al convertidor de código y decodificador, que lo transforma de formato binario a decimal para su presentación en el display o en alguna pantalla próxima a la cinta transportadora. El contenido del registro se transmite también a un multiplexor de forma que pueda ser convertido de paralelo a serie y transmitido a través de una única línea a una localización remota a cierta distancia. Es más económico instalar una única línea cuando la distancia es relativamente grande y la velocidad de los datos no es un factor importante en la aplicación que instalar varias líneas para transmitir en paralelo. En la localización remota, los datos serie son demultiplexados y enviados al registro C. A partir de este punto, los datos se decodifican para poder mostrarlos en una pantalla remota.

Téngase en mente que este sistema es un modelo ilustrativo y que no representa necesariamente ni el más actual ni el más eficiente método para implementar este hipotético proceso. Aunque efectivamente existen otros muchos métodos, se ha seleccionado éste en particular para poder ilustrar todas las funciones lógicas que se han introducido en la Sección 1.4, y que se cubrirán en detalle en los próximos capítulos.

Este ejemplo muestra una aplicación de los distintos dispositivos funcionales en el nivel de sistema y cómo pueden conectarse para cumplir un objetivo específico. En el siguiente capítulo, se verá de nuevo este sistema.

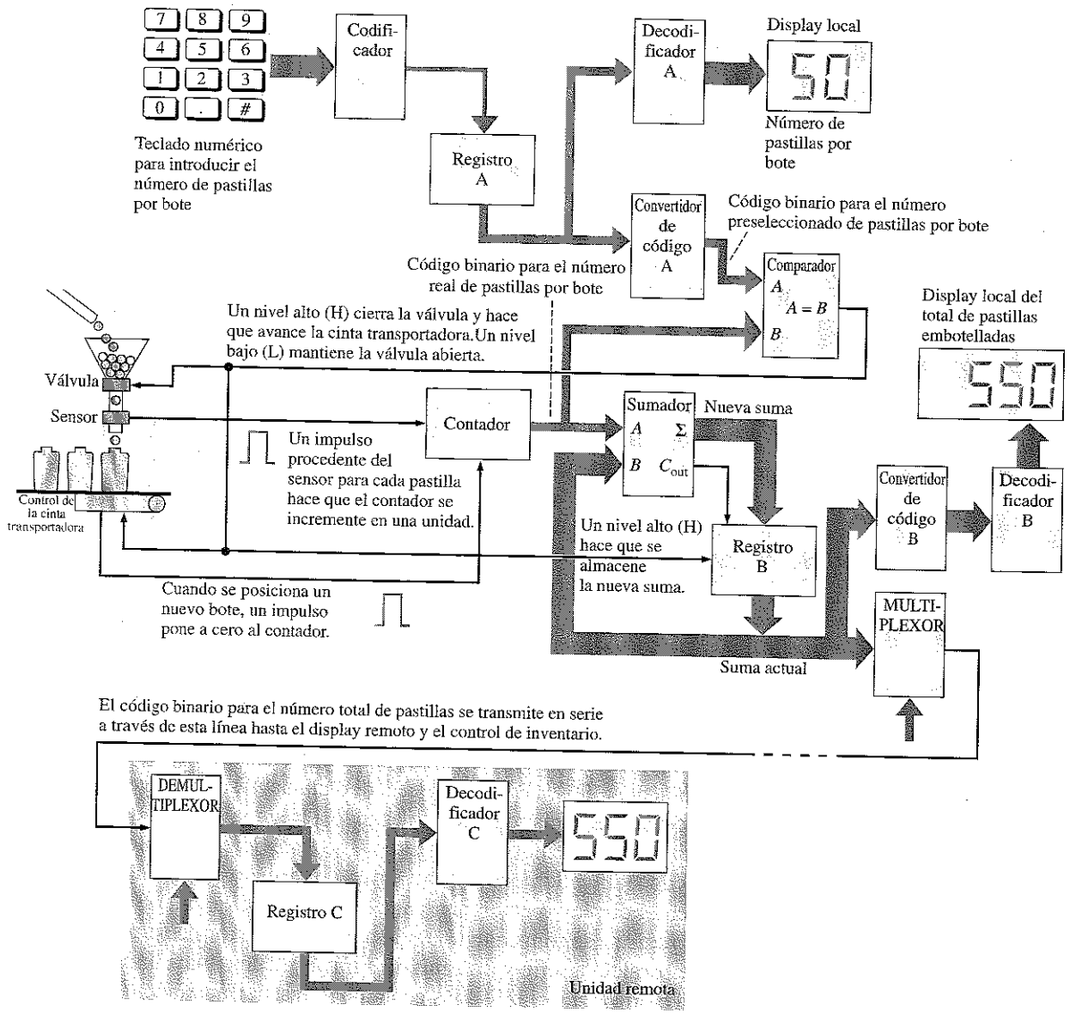


FIGURA 1.40
Diagrama de bloques simplificado de un sistema de control de recuento y envasado.

REVISIÓN DE LA SECCIÓN 1.7

1. Explicar la finalidad del comparador en el sistema de la Figura 1.40.
2. ¿Qué ocurre cuando la salida $A = B$ del comparador se pone a nivel ALTO?
3. ¿Cuál es el contenido de cada registro en cualquier instante dado?

■ RESUMEN

- Una magnitud analógica posee un conjunto continuo de valores.
- Una magnitud digital posee un conjunto discreto de valores.
- Un dígito binario se denomina bit.
- Un impulso se caracteriza por el tiempo de subida, el tiempo de bajada, el ancho del impulso y la amplitud.
- La frecuencia de una onda periódica es el recíproco del periodo. Las fórmulas que relacionan la frecuencia y el periodo son:

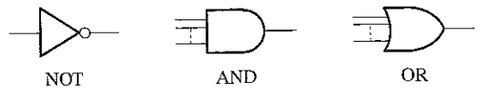
$$f = \frac{1}{T} \text{ y } T = \frac{1}{f}$$

- El ciclo de trabajo de una señal de pulsos es la relación entre el ancho del pulso y el periodo, expresado en porcentaje según la fórmula siguiente:

$$\text{Ciclo de trabajo} = \left(\frac{t_w}{T}\right)100\%$$

- Un cronograma es una representación de dos o más señales, que muestra su relación con respecto al tiempo.
- Las tres operaciones lógicas básicas son NOT, AND y OR. Sus símbolos estándar son los que se indican en la Figura 1.41.

FIGURA 1.41



- Las funciones lógicas básicas son: comparación, aritmética, conversión de código, decodificación, codificación, selección de datos, almacenamiento y recuento.
- Las dos categorías físicas más importantes de los encapsulados de CI son: montaje de inserción y montaje superficial.
- Las categorías de los CI respecto a la complejidad del circuito son: SSI (integración a baja escala), MSI (integración a escala media), LSI, VLSI y ULSI (integración a gran escala, a muy gran escala y a ultra gran escala).
- Los instrumentos más comúnmente utilizados para la comprobación y localización de averías de los circuitos digitales son: el osciloscopio, el analizador lógico, la sonda lógica, el pulsador lógico, la sonda de corriente, la fuente de alimentación DC, el generador de funciones y el multímetro digital.

■ PALABRAS CLAVE

Estos términos se encuentran también en el glosario al final del libro.

Amplitud En un tren de impulsos, la altura o máximo valor del impulso medido desde su nivel más bajo.

Analógico Que es continuo o toma valores continuos.

AND Operación lógica básica en la que se obtiene una salida verdadera (nivel ALTO) sólo si todas las condiciones de entrada son verdaderas (nivel ALTO).

Binario Que tiene dos valores o estados; describe un sistema de numeración en base dos y utiliza el 1 y el 0 como dígitos.

Bit Dígito binario que puede ser 1 o 0.

Circuito integrado (CI) Un tipo de circuito en el que todos sus componentes se encuentran integrados en un único chip semiconductor de muy pequeño tamaño.

Datos Información en formato numérico, alfabético o cualquier otro.

Digital Relacionado con los dígitos o con cantidades discretas; que posee un conjunto de valores discretos.

DIP Encapsulado dual-in-line. Un tipo de encapsulado de CI cuyos terminales pasan a través de los taladros de una tarjeta de circuito impreso hasta la otra cara.

Entrada La señal o línea que entra en un circuito.

Frecuencia El número de impulsos por segundo que hay en una onda periódica. La unidad de frecuencia es el hertzio.

Impulso Cambio brusco desde un nivel a otro, seguido, tras un cierto intervalo de tiempo, por otro cambio brusco que lo devuelve a su nivel original.

Inversor Circuito NOT. Circuito que cambia un nivel ALTO en un nivel BAJO y viceversa.

Localización de averías Técnica o proceso que consiste en identificar, aislar y corregir sistemáticamente cualquier fallo en un circuito o sistema.

Lógica En la electrónica digital, la capacidad de los circuitos de puertas de tomar decisiones, en los que un nivel ALTO representa una proposición verdadera y un nivel BAJO representa una proposición falsa.

NOT Una operación lógica básica que realiza inversiones.

OR Una operación lógica básica en la que una salida verdadera (nivel ALTO) se produce cuando una o más de las condiciones de entrada son verdaderas (nivel ALTO).

Paralelo En los circuitos digitales, datos que se producen simultáneamente a través de varias líneas. Transferencia o procesamiento de varios bits simultáneamente.

Periodo Tiempo requerido por las señales periódicas para repetirse.

Puerta Circuito lógico que realiza una operación lógica específica, tal como AND u OR.

Reloj Señal de temporización básica de un sistema digital.

Salida Señal o línea que sale de un circuito.

Serie Disponer elementos uno detrás de otro, como en una transferencia en serie de bits; en secuencia en lugar de simultáneamente.

SMT Surface-Mount Technology, tecnología de montaje superficial. Un tipo de encapsulado de circuito integrado cuyos terminales se conectan sobre la superficie de una tarjeta de circuito impreso.

Sumador Circuito lógico que se emplea para sumar dos números binarios.

Tiempo de bit El intervalo de tiempo ocupado por un único bit en una secuencia de bits; el periodo de la señal de reloj.

■ AUTOTEST

Las respuestas se encuentran al final del capítulo

1. Una magnitud que posee un conjunto continuo de valores es:
   (a) una magnitud digital **(b)** una magnitud analógica
   (c) un número binario **(d)** un número natural

2. El término *bit* significa:
   (a) una pequeña cantidad de datos **(b)** un 1 o un 0
   (c) un dígito binario **(d)** las respuestas (b) y (c)

3. El intervalo de tiempo entre el 10% y el 90% de la amplitud del flanco de subida de un impulso es el:
   (a) tiempo de subida **(b)** tiempo de bajada
   (c) ancho del impulso **(d)** periodo

4. Un impulso de una cierta señal se produce cada 10 ms. La frecuencia es:
   (a) 1 kHz **(b)** 1 Hz **(c)** 100 Hz **(d)** 10 Hz

5. En una cierta señal digital, el periodo es el doble del ancho del impulso. El ciclo de trabajo es:
   (a) 100% **(b)** 200% **(c)** 50%

6. Un inversor:
   (a) realiza la operación NOT **(b)** cambia un nivel ALTO a un nivel BAJO
   (c) cambia un nivel BAJO a un nivel ALTO **(d)** todas las anteriores

7. La salida de una puerta AND se pone a nivel ALTO cuando:
   (a) cualquier entrada está a nivel ALTO **(b)** todas las entradas están a nivel ALTO
   (c) ninguna entrada está a nivel ALTO **(d)** las respuestas (a) y (b)

8. La salida de una puerta OR se pone a nivel ALTO cuando:
   (a) cualquier entrada está a nivel ALTO **(b)** todas las entradas están a nivel ALTO
   (c) ninguna entrada está a nivel ALTO **(d)** a y b

9. El dispositivo utilizado para convertir un número binario en un formato para display de 7 segmentos es:
   (a) multiplexor **(b)** codificador **(c)** decodificador **(d)** registro

10. Un ejemplo de dispositivo de almacenamiento es:
    (a) la puerta lógica **(b)** el flip-flop
    (c) el comparador **(d)** el registro **(e)** las respuestas (b) y (d)

11. Un CI que contiene cuatro puertas AND es un ejemplo de:
    (a) MSI **(b)** SMT **(c)** SOIC **(d)** SSI

12. Un dispositivo LSI tiene una complejidad de:
    (a) 12 a 99 puertas equivalentes
    (b) 100 a 9.999 puertas equivalentes
    (c) 2.000 a 5.000 puertas equivalentes
    (d) 10.000 a 99.999 puertas equivalentes

■ PROBLEMAS

Las respuestas a la mayoría de los problemas impares se encuentran al final del libro.

SECCIÓN 1.1 Magnitudes digitales y analógicas

1. Citar dos ventajas de los datos digitales en comparación con los analógicos.
2. Citar una magnitud analógica que no sea la temperatura o el sonido.

SECCIÓN 1.2 Dígitos binarios, niveles lógicos y formas de onda digitales

3. Definir la secuencia de bits (unos 1 y ceros 0) representados por cada una de las siguientes secuencias de niveles.

(a) ALTO, ALTO, BAJO, ALTO, BAJO, BAJO, BAJO, ALTO.
(b) BAJO, BAJO, BAJO, ALTO, BAJO, ALTO, BAJO, ALTO, BAJO.

4. Enumerar la secuencia de niveles (ALTO y BAJO) que representan cada una de las siguientes secuencias de bits:

(a) 1011101 **(b)** 11101001

5. Para el impulso mostrado en la Figura 1.42 determinar gráficamente:

(a) el tiempo de subida **(b)** el tiempo de bajada **(c)** el ancho del impulso **(d)** la amplitud

6. Determinar el periodo de la señal digital de la Figura 1.43.
7. ¿Cuál es la frecuencia de la señal de la Figura 1.43?
8. ¿El tren de impulsos de la Figura 1.43 es periódico o aperiódico?
9. Determinar el ciclo de trabajo de la señal de la Figura 1.43.
10. Determinar la secuencia de bits representada por la señal de la Figura 1.44. En este caso, el periodo de bit es de 1 μs.
11. ¿Cuál es el tiempo total de transmisión serie de los ocho bits de la Figura 1.44? ¿Cuál es el tiempo total para la transmisión en paralelo?

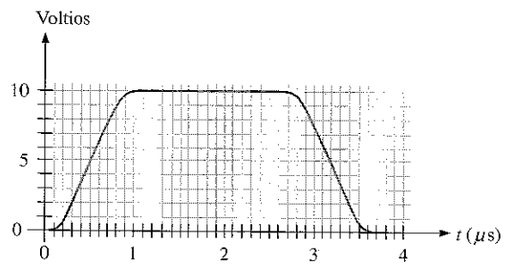


FIGURA 1.42

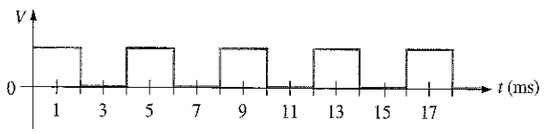
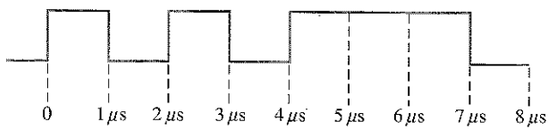


FIGURA 1.43

FIGURA 1.44



SECCIÓN 1.3 Operaciones lógicas básicas

12. Un circuito lógico requiere un nivel ALTO en todas sus entradas para conseguir un nivel ALTO de salida. ¿Qué tipo de circuito lógico es?
13. Un circuito lógico básico de 2 entradas tiene una entrada a nivel ALTO y la otra a nivel BAJO, la salida está a nivel BAJO. Identificar el circuito.
14. Un circuito lógico básico de 2 entradas tiene una entrada a nivel ALTO y la otra a nivel BAJO, y la salida está a nivel ALTO. ¿Qué tipo de circuito lógico es?

SECCIÓN 1.4 Funciones lógicas básicas

15. Identificar la función de cada bloque de la Figura 1.45 en función de las entradas y salidas.

16. Un tren de impulsos con una frecuencia de 10 kHz se aplica a la entrada de un contador. En 100 ms, ¿cuántos impulsos se contarán?

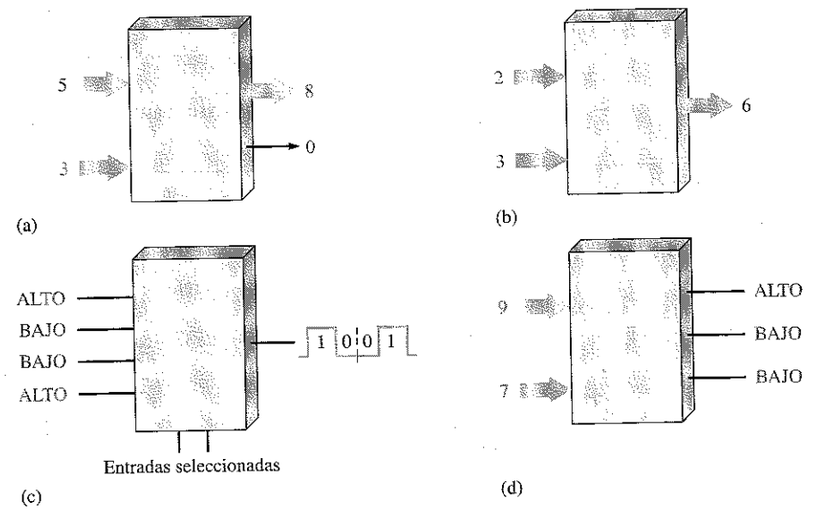


FIGURA 1.45

17. Considerar un registro que puede almacenar 8 bits. Suponer que se ha puesto a cero para que contenga ceros en todas sus posiciones. Si transferimos cuatro bits alternativos (0101) en modo serie al registro, comenzando con un 1 y desplazándolo a la derecha, ¿cuál será el contenido del registro cuando se almacena el cuarto bit?

SECCIÓN 1.5 Circuitos integrados digitales

18. Un chip tiene una complejidad de 200 puertas equivalentes. ¿Cómo se clasifica?
19. Explicar la diferencia principal entre los encapsulados DIP y SMT.
20. Numerar los pines de los encapsulados de la Figura 1.46. Se muestra las vistas frontales.

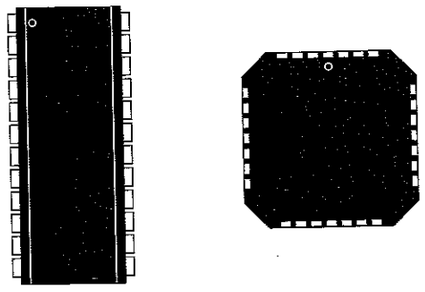


FIGURA 1.46 (a) (b)

SECCIÓN 1.6 Instrumentación para localización de averías y pruebas

21. Se observa un impulso en la pantalla de un osciloscopio y medimos la línea base en 1 V y el máximo del impulso en 8 V. ¿Cuál es la amplitud?

22. Una sonda lógica se aplica a un punto de contacto sobre un CI que está operando en un sistema. La luz de la sonda lógica parpadea repetidamente. ¿Qué indica?

SECCIÓN 1.7 Aplicación a los sistemas digitales

23. Definir el término *sistema*.
24. En el sistema mostrado en la Figura 1.40, ¿para qué son necesarios el multiplexor y el demultiplexor?
25. ¿Qué podemos hacer para cambiar el número de pastillas por bote en el sistema de la Figura 1.40?

■ RESPUESTAS A LAS REVISIONES DE CADA SECCIÓN

SECCIÓN 1.1

1. Analógico significa continuo.
2. Digital significa discreto.
3. Una magnitud digital posee un conjunto discreto de valores y una magnitud analógica posee un conjunto continuo de valores.
4. Un altavoz es analógico. Un reproductor de CD es analógico y digital. Una computadora es completamente digital.

SECCIÓN 1.2

1. Binario significa que tiene dos estados o valores.
2. Un bit es un dígito binario.
3. Los bits son 1 y 0.
4. Tiempo de subida: del 10% al 90% de la amplitud. Tiempo de bajada: del 90% al 10% de la amplitud.
5. La frecuencia es el recíproco del periodo.
6. Una señal de reloj es la señal de temporización básica de la que se derivan las demás.
7. Un diagrama de tiempos muestra las relaciones de tiempo entre señales.
8. La transferencia paralelo es más rápida que la transferencia serie.

SECCIÓN 1.3

1. Cuando la entrada es un nivel BAJO.
2. Cuando todas las entradas están a nivel ALTO.
3. Cuando cualquier entrada o todas están a nivel ALTO.
4. Un inversor es un circuito NOT.
5. Una puerta lógica es un circuito que realiza una operación lógica (AND, OR).

SECCIÓN 1.4

1. Un comparador compara las magnitudes de dos números de entrada.
2. Sumar, restar, multiplicar y dividir.
3. Codificar es cambiar un formato familiar, como decimal, a un formato codificado, como el binario.
4. Decodificar es cambiar un código a un formato familiar, como de binario a decimal.
5. Multiplexar es poner datos procedentes de muchas fuentes en una única línea. Demultiplexar es tomar datos de una única línea y distribuirlos a muchos destinos.
6. Flip-flops, registros, memorias semiconductoras, discos magnéticos.
7. Un contador cuenta sucesos mediante una secuencia de estados binarios.

SECCIÓN 1.5

1. Un CI es un circuito electrónico con todos los componentes integrados en un único chip de silicio.
2. DIP - Encapsulado dual-in-line. SMT - Tecnología de montaje superficial. SOIC - CI de perfil BAJO. SSI - Integración a pequeña escala. MSI - Integración a media escala. LSI - Integración a gran escala. VLSI - Integración a muy gran escala. ULSI - Integración a ultra gran escala.
3. **(a)** MSI **(b)** LSI **(c)** SSI **(d)** VLSI **(e)** ULSI

SECCIÓN 1.6

1. Diez.
2. Ocho.
3. Analizador lógico.
4. Indica impulsos repetitivos.

SECCIÓN 1.7

1. El comparador determina cuándo el número de pastillas es igual al número preseleccionado de pastillas por bote.
2. La válvula de suministro se cierra, el siguiente bote se mueve sobre la cinta, y la nueva suma se almacena en el registro B.
3. El registro A almacena el número preseleccionado de pastillas por bote. El registro B almacena el número total de pastillas envasadas.

■ RESPUESTAS A LOS PROBLEMAS RELACIONADOS DE LOS EJEMPLOS

1.1 $f = 6{,}67$ kHz; ciclo de trabajo = 16,7%
1.2 Transferencia paralelo: 100 ns; Transferencia serie = 1,6 µs
1.3 Amplitud = 12 V; $T = 8$ ms

■ RESPUESTAS AL AUTOTEST

1. (b) **2.** (d) **3.** (a) **4.** (c) **5.** (c) **6.** (d) **7.** (b)
8. (d) **9.** (c) **10.** (e) **11.** (d) **12.** (d)

2

SISTEMAS DE NUMERACIÓN, OPERACIONES Y CÓDIGOS

2.1 Números decimales
2.2 Números binarios
2.3 Conversión decimal-binario
2.4 Aritmética binaria
2.5 Complemento a 1 y complemento a 2 de los números binarios
2.6 Números con signo
2.7 Operaciones aritméticas de números con signo
2.8 Números hexadecimales
2.9 Números octales
2.10 Código decimal binario (BCD)
2.11 Códigos digitales y paridad
2.12 Aplicación a los sistemas digitales

Simulación por computadora de circuitos digitales:
El tutorial de Electronics Workbench (EWB) se encuentra en la dirección http://www.prenhall.com/floyd

■ OBJETIVOS DEL CAPÍTULO

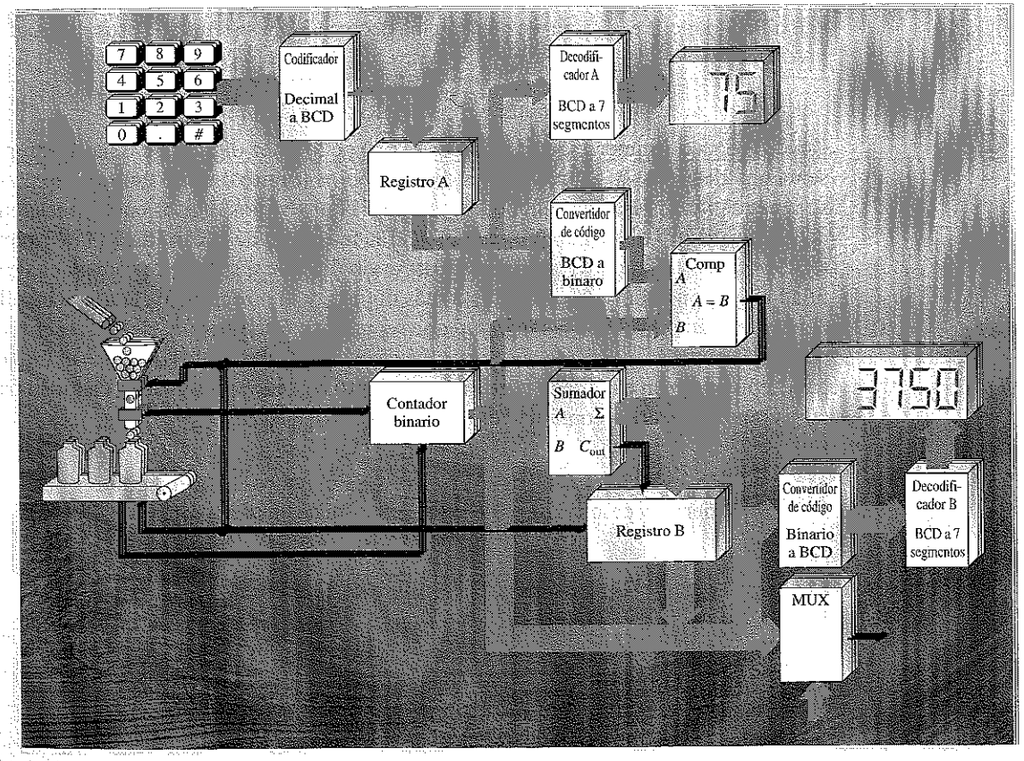
- ❑ Revisión del sistema numérico decimal.
- ❑ Contar en el sistema de numeración binario.
- ❑ Convertir de decimal a binario y de binario a decimal.
- ❑ Aplicar las operaciones aritméticas a los números binarios.
- ❑ Determinar el complemento a 1 y el complemento a 2 de un número binario.
- ❑ Expresar números con signo en los formatos binarios: signo-magnitud, complemento a 1, complemento a 2 y coma flotante.
- ❑ Realizar operaciones aritméticas con números binarios con signo.
- ❑ Conversión entre los sistemas de numeración hexadecimal y binario.
- ❑ Sumar números en formato hexadecimal.
- ❑ Conversión entre los sistemas de numeración binario y octal.
- ❑ Expresar los números decimales en formato BCD (Binary Coded Decimal, código decimal binario).
- ❑ Sumar números en BCD.
- ❑ Conversión entre el sistema binario y el código Gray.
- ❑ Interpretar el código ASCII (American Standard Code for Information Interchange, código estándar americano para el intercambio de información)
- ❑ Utilización de los códigos y números binarios en una aplicación de sistemas.

■ INTRODUCCIÓN

El sistema de numeración binario y los códigos digitales son fundamentales en las computadoras y, en general, en la electrónica digital. Este capítulo está enfocado principalmente al sistema de numeración binario y sus relaciones con otros sistemas de numeración tales como el decimal, hexadecimal y octal. Se cubren las operaciones aritméticas con números binarios, con el fin de proporcionar una base para entender cómo trabajan las computadoras y muchos otros tipos de sistemas digitales. También se abordan códigos digitales tales como el código decimal binario (Binary Coded Decimal, BCD), el código Gray y el código ASCII. Se introduce el método de paridad para la detección de errores en los códigos. Las sugerencias sobre el uso de la calculadora en determinadas operaciones están basadas en las calculadoras TI-85 y TI-86. Los procedimientos mostrados pueden variar en otros tipos de calculadoras.

■ APLICACIÓN A LOS SISTEMAS DIGITALES

Esta aplicación a un sistema digital ilustra los conceptos que se enseñan en este capítulo. Para mostrar cómo se usan en una aplicación los códigos y números binarios, se usa como ejemplo el sistema de control y contador de pastillas que se ha introducido en el Capítulo 1. En la Sección 2.12, "Aplicaciones a los sistemas digitales", se examinarán los códigos y números binarios en varios puntos del sistema y se verá cómo pasa el sistema a través de un ciclo de recuento. Se analizará el sistema para determinar los estados binarios para ciertas condiciones específicas.



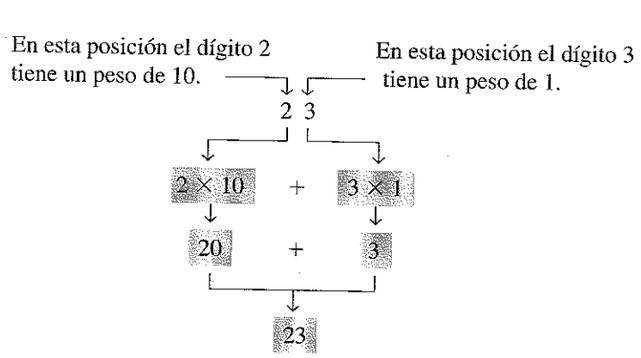
2.1 ■ NÚMEROS DECIMALES

Todos estamos familiarizados con el sistema de numeración decimal porque usamos los números decimales cada día. Aunque los números decimales son triviales, a menudo, su estructura de pesos no se comprende. En esta sección, revisaremos la estructura de los números decimales. Esta revisión le ayudará a entender más fácilmente la estructura del sistema de numeración binario, que es tan importante en las computadoras y la electrónica digital.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar por qué el sistema de numeración decimal es un sistema de pesos. ■ Definir *potencias de diez*. ■ Determinar el peso de cada dígito en un número decimal.

En el sistema de numeración **decimal** cada uno de los diez dígitos, de 0 hasta 9, representa una determinada cantidad. Los diez símbolos (**dígitos**) no se limitan a expresar solamente diez diferentes cantidades, ya que usamos varios dígitos en las posiciones adecuadas dentro de un número para indicar la magnitud de la cantidad. Puede expresar cantidades hasta nueve antes de quedarse sin dígitos: si se desea expresar una cantidad mayor que nueve, se usan dos o más dígitos, y la posición de cada uno de ellos dentro del número le dice la magnitud que representa. Por ejemplo, si desea escribir la cantidad veintitrés, usará (en sus respectivas posiciones dentro del número) el dígito 2 para representar la cantidad de veinte, y el dígito 3 para representar la cantidad tres, tal y como se ilustra a continuación



La posición de cada dígito en un número decimal indica la magnitud de la cantidad representada, y se le puede asignar un **peso.** Los pesos para los números enteros son potencias positivas de diez, que aumentan de derecha a izquierda, comenzando por $10^0 = 1$.

$$\ldots 10^5\ 10^4\ 10^3\ 10^2\ 10^1\ 10^0$$

Para números fraccionarios, los pesos son potencias negativas de diez que aumentan de izquierda a derecha, comenzando por 10^{-1}.

$$10^2\ \ 10^1\ \ 10^0, 10^{-1}\ \ 10^{-2}\ \ 10^{-3}\ \ldots$$
$$\uparrow$$
Coma decimal

El valor de un número decimal es la suma de los dígitos después de haber multiplicado cada dígito por su peso, como ilustran los ejemplos 2.1 y 2.2.

EJEMPLO 2.1 Expresar el número decimal 47 como suma de los valores de cada dígito.

Solución. Como indican sus respectivas posiciones, el dígito 4 tiene un peso de 10, que es 10^1. El dígito 7 tiene un peso de 1, que corresponde a 10^0.

$$\begin{aligned} 47 &= (4 \times 10^1) + (7 \times 10^0) \\ &= (4 \times 10) + (7 \times 1) = \mathbf{40 + 7} \end{aligned}$$

Problema relacionado. Determinar el valor de cada dígito en el número 939.

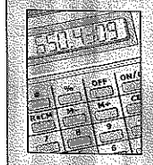
EJEMPLO 2.2 Expresar el número decimal 568,23 como suma de los valores de cada dígito.

Solución. El dígito 5 de la parte entera del número tiene un peso de 100, es decir 10^2; el dígito 6 tiene un peso de 10, que corresponde a 10^1; el dígito 8 tiene un peso de 1, que es 10^0; el dígito 2 de la parte fraccionaria tiene un peso de 0,1, es decir 10^{-1}; y el dígito 3 tiene un peso de 0,01, que es 10^{-2}.

$$\begin{aligned} 568{,}23 &= (5 \times 10^2) + (6 \times 10^1) + (8 \times 10^0) + (2 \times 10^{-1}) + (3 \times 10^{-2}) \\ &= (5 \times 100) + (6 \times 10) + (8 \times 1) + (2 \times 0{,}1) + (3 \times 0{,}01) \\ &= \mathbf{500 + 60 + 8 + 0{,}2 + 0{,}03} \end{aligned}$$

Problema relacionado. Determinar el valor de cada dígito en el número 67,924.

Consejos para el uso de la calculadora



Las potencias de diez se pueden calcular con la calculadora* empleando la tecla 10^x.

Ejemplo. Calcular el valor de 10^3.

Paso 1. Presionar [2nd] y luego [10^x].

Paso 2. Presionar [3] y luego [ENTER].

Display

10^3
1000 ← *Resultado*

* En estos consejos se ha usado la calculadora TI-85.

REVISIÓN DE LA SECCIÓN 2.1

1. ¿Qué peso tiene el dígito 7 en cada uno de los siguientes números?
   (a) 1370 **(b)** 6725 **(c)** 7051 **(d)** 58,72
2. Expresar cada uno de los siguientes números decimales como una suma de productos obtenida mediante la multiplicación de cada dígito por su peso correspondiente:
   (a) 51 **(b)** 137 **(c)** 1492 **(d)** 106,58

2.2 ■ NÚMEROS BINARIOS

El sistema de numeración binario es simplemente otra forma de representar magnitudes. El sistema binario es menos complicado que el sistema decimal ya que sólo tiene dos dígitos. Al principio puede parecer más complicado por no ser familiar. El sistema decimal con sus diez dígitos es un sistema en base diez, el sistema binario con sus dos dígitos es un sistema en base dos. Los dos dígitos binarios (bits) son 1 y 0. La posición de un 1 o un 0 en un número binario indica su peso, o valor dentro del número, así como la posición de un dígito decimal determina el valor de ese dígito. Los pesos de un número binario están basados en las potencias de dos.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Contar en binario. ■ Determinar el mayor número decimal que se puede representar con un número dado de bits. ■ Convertir un número binario en un número decimal.

Contar en binario

Para aprender a contar en el sistema **binario**, en primer lugar, hay que observar cómo contamos en el sistema decimal. Empezamos en cero y contamos hasta nueve antes de quedarnos sin dígitos. Luego, comenzamos con otra posición de dígito (a la izquierda) y contamos desde 10 hasta 99. En este punto, se terminan todas las combinaciones con dos dígitos, por lo que es necesaria una tercera posición de dígito para contar desde 100 hasta 999.

Cuando contamos en binario se produce una situación comparable, excepto que ahora sólo tenemos dos dígitos, llamados **bits**. Empecemos a contar: 0, 1. Hasta aquí, ya hemos usado los dos dígitos, por lo que incluimos otra posición de dígito y continuamos: 10, 11. Ahora, hemos agotado todas las combinaciones de dos dígitos, por lo que se requiere una tercera posición. Con tres dígitos podemos continuar contando: 100, 101, 110 y 111. Para continuar, necesitamos una cuarta posición de dígito, y así sucesivamente. En la Tabla 2.1 se muestra cómo se cuenta de cero hasta quince. Observe la alternancia de 1s y 0s en cada columna.

Como puede verse en la Tabla 2.1, se requieren cuatro bits para contar desde cero hasta 15. En general, con n bits se puede contar hasta un número igual a 2^n-1.

$$\text{Máximo número decimal} = 2^n\text{-}1$$

Por ejemplo, con cinco bits ($n = 5$) se puede contar desde 0 hasta treinta y uno.

$$2^5\text{–}1 = 32\text{–}1 = 31$$

TABLA 2.1

Número decimal	Número binario			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Con seis bits ($n = 6$) se puede contar desde 0 hasta sesenta y tres.

$$2^6\text{–}1 = 64 - 1 = 63$$

En el Apéndice B se facilita una tabla con las potencias de dos.

Consejos para el uso de la calculadora

Las potencias de dos se pueden obtener en la calculadora utilizando la tecla [^].

Ejemplo. Calcular el valor de 2^5

Paso 1. Presionar [2] y luego [^].

Paso 2. Presionar [5] y luego [ENTER].

Display

```
2 ^ 5
      32
```

Ejemplo de aplicación

Aprender a contar en binario le ayudará a entender básicamente cómo pueden utilizarse los circuitos digitales para contar sucesos. Puede tratarse de cualquier cosa, desde elementos que contar en una línea de montaje hasta operaciones de recuento en una computadora. Tomemos un sencillo ejemplo para contar las pelotas de tenis que se desplazan por una cinta transportadora hasta meterse en una caja. Supongamos que en cada caja se introducen nueve pelotas.

El contador mostrado en la Figura 2.1 cuenta los pulsos procedentes de un sensor que detecta el paso de una pelota y genera una secuencia de niveles lógicos (ondas digitales), en cada una de sus cuatro salidas paralelas. Cada conjunto de niveles lógicos representa un número binario de 4 bits (ALTO (H) = 1 y BAJO (L) = 0), como se indica. Cuando el decodificador recibe estas formas de onda, decodifica cada conjunto de 4 bits y lo convierte en el correspondiente número decimal en el display de siete segmentos. Cuando el contador alcanza el estado binario 1001, ha contado nueve pelotas de tenis, y el display muestra el decimal 9, una nueva caja se desplaza por la cinta transportadora, el contador se pone de nuevo en su estado cero (0000) y el proceso comienza de nuevo. El número 9 sólo se ha utilizado en interés de la simplicidad que ofrece un único dígito.

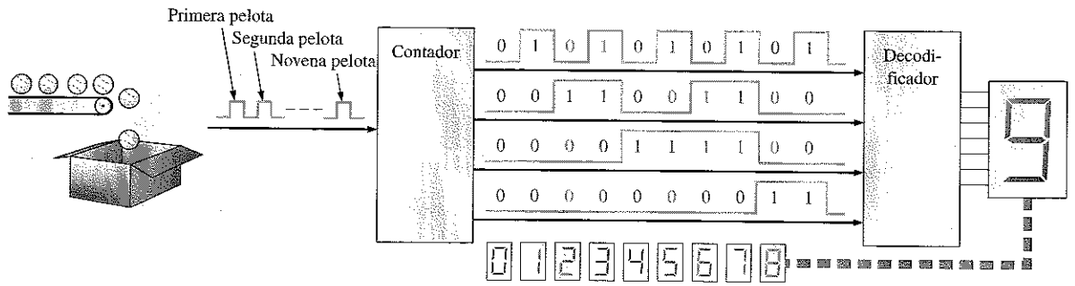


FIGURA 2.1
Ilustración de una aplicación de recuento binario.

NOTAS INFORMÁTICAS
Las computadoras utilizan los números binarios para seleccionar posiciones de memoria. Cada posición se asigna a un único número denominado *dirección*. Por ejemplo, el microprocesador Pentium tiene 32 líneas de dirección que pueden seleccionar 2^{32} (4.294.967.296) posiciones unívocas.

La estructura de pesos de los números binarios

Un número binario es un número con peso. El bit más a la derecha es el **bit menos significativo (LSB**, Least Significant Bit) en un número entero binario y tiene un peso de $2^0 = 1$. Los pesos de los respectivos bits crecen de derecha a izquierda según las potencias de dos. El bit más a la izquierda es el **bit más significativo (MSB**, Most Significant Bit), y su peso depende del tamaño del número binario.

Los números con parte fraccionaria también se pueden representar en binario, colocando bits a la derecha del punto binario, del mismo modo que los dígitos decimales fraccionarios se colocan a la derecha del punto decimal. En un número binario con parte fraccionaria, el bit más a la izquierda es el MSB, y tiene un peso de $2^{-1} = 0{,}5$. Los pesos fraccionarios de los respectivos bits decrecen de izquierda a derecha según las potencias negativas de dos.

La estructura de pesos de un número binario es:

$$2^{n-1} \ldots 2^3\ 2^2\ 2^1\ 2^0, 2^{-1}\ 2^{-2} \ldots 2^{-n}$$
$$\uparrow \text{ Coma binaria}$$

donde n es el número de bits a partir del punto binario. Luego todos los bits a la izquierda de la coma binaria tienen pesos que son potencias positivas de dos, como previamente se estableció. Todos los bits a la derecha de la coma binaria tienen pesos que son potencias negativas de dos, o pesos fraccionales.

En la Tabla 2.2 se muestran las potencias de dos y su equivalente decimal de un número entero binario de 8 bits y de un número fraccionario binario de 6 bits. Obsérvese que el peso se duplica para cada potencia de dos y se divide por dos para cada potencia negativa de dos. Puede ampliar fácilmente esta tabla duplicando el peso de la potencia positiva de dos más significativa y dividiendo por dos el peso de la potencia negativa de dos menos significativa; por ejemplo, $2^9 = 512$ y $2^{-7} = 0{,}0078125$.

TABLA 2.2
Pesos binarios

Potencias positivas de dos (número entero)									Potencias negativas de dos (número fraccionario)					
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
256	128	64	32	16	8	4	2	1	1/2 0,5	1/4 0,25	1/8 0,125	1/16 0,0625	1/32 0,03125	1/64 0,015625

Conversión binario a decimal

El valor decimal de cualquier número binario se puede determinar sumando los pesos de todos los bits que son 1, y descartando los pesos de todos los bits que son 0. Los dos ejemplos siguientes ilustran esto.

EJEMPLO 2.3 Convertir el número entero binario 1101101 a decimal.

Solución. Se determina el peso de cada bit que está a 1, y luego se obtiene la suma de los pesos para obtener el número decimal.

Peso: $2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$
Número binario: 1 1 0 1 1 0 1

$$1101101 = 2^6 + 2^5 + 2^3 + 2^2 + 2^0$$
$$= 64 + 32 + 8 + 4 + 1 = \mathbf{109}$$

Problema relacionado. Convertir el número binario 10010001 a decimal.

EJEMPLO 2.4 Convertir el número binario fraccionario 0,1011 en decimal.

Solución. En primer lugar, se determina el peso de cada bit que está a 1, y luego se suman los pesos para obtener la fracción decimal.

Peso: $2^{-1}\ 2^{-2}\ 2^{-3}\ 2^{-4}$
Número binario: 0, 1 0 1 1

$$0.1011 = 2^{-1} + 2^{-3} + 2^{-4}$$
$$= 0{,}5 + 0{,}125 + 0{,}0625 = \mathbf{0{,}6875}$$

Problema relacionado. Evaluar el número binario 10,111 en decimal.

REVISIÓN DE LA SECCIÓN 2.2

1. ¿Cuál es el mayor número decimal que se puede representar en binario con ocho bits?
2. Determinar el peso del 1 en el número binario 10000.
3. Convertir el número binario 10111101,011 a decimal.

2.3 ■ CONVERSIÓN DECIMAL-BINARIO

En la Sección 2.2 hemos estudiado cómo convertir un número binario en el número decimal equivalente. Ahora vamos a aprender dos métodos para convertir un número decimal en un número binario.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Convertir a binario un número decimal utilizando el método de la suma de pesos. ■ Convertir a binario un número decimal entero utilizando el método de la división sucesiva por dos. ■ Convertir a binario un número decimal con parte fraccionaria utilizando el método de la multiplicación sucesiva por dos.

Método de la suma de pesos

Una forma de calcular el número binario equivalente a un número decimal dado es determinar el conjunto de pesos binarios, cuya suma es igual al número decimal. Una forma fácil de recordar los pesos binarios es que el peso más bajo es 1, es decir 2^0, y que duplicando cualquier peso, se obtiene el siguiente peso superior; por tanto, la lista de los siete primeros pesos binarios sería: 1, 2, 4, 8, 16, 32, 64, como se aprenderá en la última sección. El número decimal 9, por ejemplo, puede expresarse como la suma de los pesos binarios siguientes:

$$9 = 8 + 1 \quad \text{o} \quad 9 = 2^3 + 2^0$$

Colocando los 1s en las posiciones de pesos adecuadas, 2^3 y 2^0, y los 0s en las posiciones 2^2 y 2^1, se determina el número binario correspondiente al decimal 9:

$2^3\ 2^2\ 2^1\ 2^0$
1 0 0 1 Nueve en binario

EJEMPLO 2.5 Convertir los siguientes números decimales a formato binario:

(a) 12 **(b)** 25 **(c)** 58 **(d)** 82

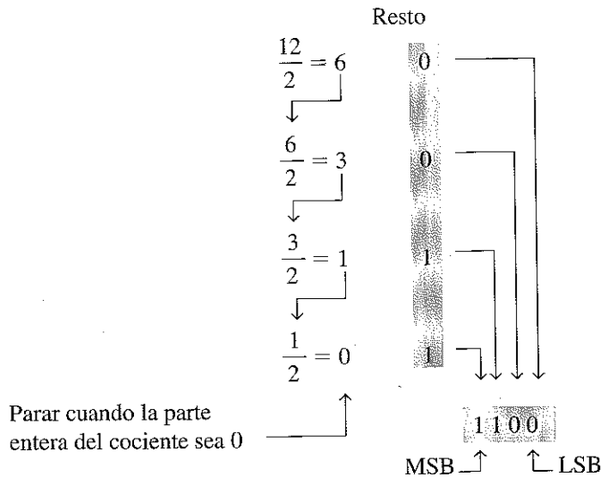
Solución

(a) $12 = 8 + 4 = 2^3 + 2^2 \longrightarrow$ **1100**
(b) $25 = 16 + 8 + 1 = 2^4 + 2^3 + 2^0 \longrightarrow$ **11001**
(c) $58 = 32 + 16 + 8 + 2 = 2^5 + 2^4 + 2^3 + 2^1 \longrightarrow$ **111010**
(d) $82 = 64 + 16 + 2 = 2^6 + 2^4 + 2^1 \longrightarrow$ **1010010**

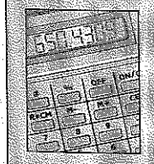
Problema relacionado. Convertir a binario el número decimal 125.

Método de la división sucesiva por 2

Un método sistemático para convertir a binario números enteros decimales es el proceso de la *división sucesiva por 2*. Por ejemplo, para convertir a binario el número decimal 12, comenzamos dividiendo 12 entre 2. Luego cada cociente resultante se divide por 2 hasta que se obtiene un cociente cuya parte entera es 0. Los **restos** generados en cada división forman el número binario. El primer resto es el bit menos significativo (LSB) del número binario, y el último resto es el bit más significativo (MSB). Este procedimiento, se muestra en los pasos siguientes para convertir el número decimal 12 en binario.



Consejos para el uso de la calculadora



Un número decimal puede convertirse en un número binario utilizando la secuencia de teclas de conversión 2nd BASE F3.

Ejemplo. Conversión a binario del número decimal 57.

Paso 1. Presionar 5 7. 57 ▶Bin

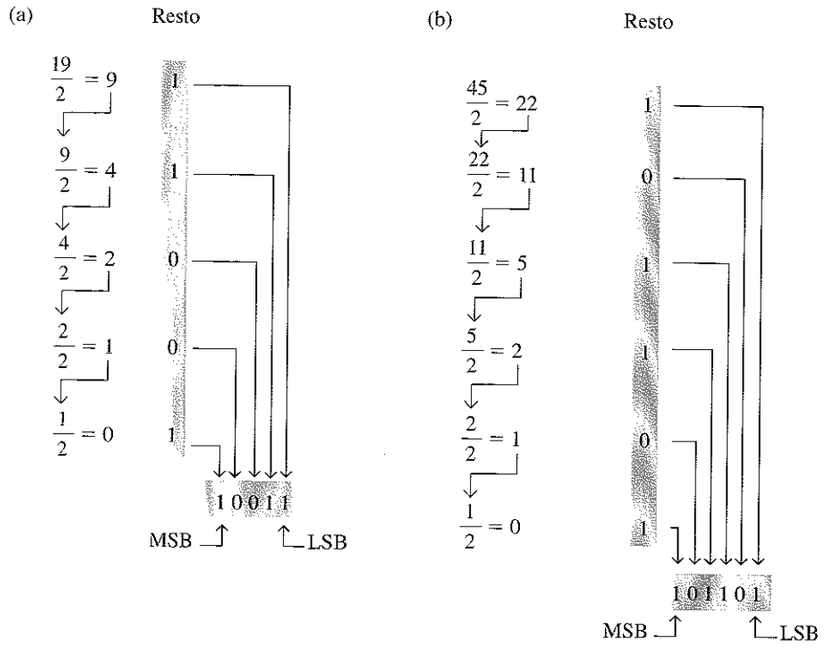
Paso 2. Presionar F1. 111001b

Paso 3. Presionar ENTER.

EJEMPLO 2.6 Convertir los siguientes números decimales en números binarios:

(a) 19 **(b)** 45

Solución



Problema relacionado. Convertir a binario el número decimal 39.

Conversión de fracciones decimales a binario

En los Ejemplos 2.5 y 2.6 se han mostrado conversiones de números enteros. Ahora, vamos a ver la conversión de números fraccionarios. Una forma fácil de recordar los pesos binarios fraccionarios es que el peso más significativo es 0,5, es decir 2^{-1}, y que dividiendo por dos cualquier peso, se puede obtener el siguiente peso menor; luego una lista para los cuatro primeros pesos binarios fraccionarios sería 0,5; 0,25; 0,125; 0,0625.

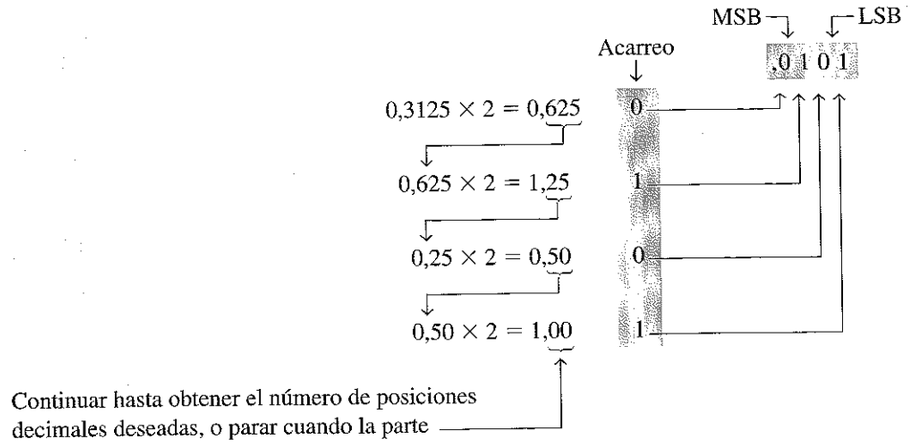
Suma de pesos. El método de la suma de pesos se puede aplicar a números decimales fraccionarios, como se muestra en el siguiente ejemplo:

$$0{,}625 = 0{,}5 + 0{,}125 = 2^{-1} + 2^{-3} = 0{,}101$$

Lo que indica que en la posición 2^{-1} hay un 1, en la posición 2^{-2} un 0 y en la posición 2^{-3} un 1.

Multiplicación sucesiva por 2. Como hemos visto, los números decimales enteros se pueden convertir a números binarios mediante la división sucesiva por 2. Los números decimales fraccionarios pueden convertirse en números binarios mediante la multiplicación sucesiva por 2. Por ejemplo, para convertir a binario el número decimal fraccionario 0,315, empezamos multiplicando por 2, y después se multiplica cada

parte fraccional resultante del producto por 2, hasta que el producto fraccionario sea cero o hasta que se alcance el número deseado de posiciones decimales. Los dígitos acarreados, o **acarreos**, generados por las multiplicaciones dan lugar al número binario. El primer acarreo que se obtiene es el MSB, y el último acarreo es el LSB. Este procedimiento se ilustra como sigue:



REVISIÓN DE LA SECCIÓN 2.3

1. Convertir a binario los siguientes números decimales utilizando el método de la suma de pesos:
   (a) 23 **(b)** 57 **(c)** 45,5
2. Convertir a binario los siguientes números decimales utilizando el método de la división sucesiva por 2 (multiplicación sucesiva por 2 para números fraccionarios):
   (a) 14 **(b)** 21 **(c)** 0,375

2.4 ■ ARITMÉTICA BINARIA

La aritmética binaria es esencial en todas las computadoras digitales y en muchos otros tipos de sistemas digitales. Para entender los sistemas digitales, debe conocer los principios básicos de la suma, sustracción, multiplicación y división binarias. En esta sección se hace una introducción, que será ampliada en las secciones posteriores.

Al finalizar esta sección, el lector deberá ser capaz de:

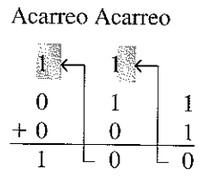
■ Sumar números binarios. ■ Restar números binarios. ■ Multiplicar números binarios. ■ Dividir números binarios.

Suma binaria

Las cuatro reglas básicas para sumar dígitos binarios son:

0 + 0 = 0	Suma 0 con acarreo 0
0 + 1 = 1	Suma 1 con acarreo 0
1 + 0 = 1	Suma 1 con acarreo 0
1 + 1 = 10	Suma 0 con acarreo 1

Observe que las tres primeras reglas dan lugar a un resultado de un único bit, y la cuarta regla, la suma de dos 1s, da lugar a 2 en binario (10). Cuando se suman números binarios, teniendo en cuenta la última regla se obtiene en la columna dada la suma 0 y un acarreo de 1 que pasa a la siguiente columna de la izquierda, tal y como se muestra en la siguiente suma de 11 +1:



En la columna de la derecha 1 + 1 = 0 con acarreo 1, que pasa a la siguiente columna de la izquierda. En la columna central, 1 + 1 + 0 = 0 con acarreo 1, que pasa a la siguiente columna de la izquierda. Y en la columna de la izquierda, 1 + 0 + 0 = 1.

Cuando existe un acarreo igual a 1, se produce una situación en la que se deben sumar tres bits (un bit de cada uno de los números y un bit de acarreo). Esta situación se ilustra como sigue:

1 + 0 + 0 = 01	suma 1 con acarreo 0
1 + 1 + 0 = 10	suma 0 con acarreo 1
1 + 0 + 1 = 10	suma 0 con acarreo 1
1 + 1 + 1 = 11	suma 1 con acarreo 1

El Ejemplo 2.7 ilustra la suma binaria.

EJEMPLO 2.7 Sumar los siguientes números binarios:
(a) 11 + 11 **(b)** 100 + 10 **(c)** 111 + 11 **(d)** 110 + 100

Solución. La suma decimal equivalente se muestra también como referencia.

(a) 11	3	**(b)** 100	4	**(c)** 111	7	**(d)** 110	6
+ 11	+ 3	+ 10	+ 2	+ 1	+ 3	+ 100	+ 4
110	6	**110**	6	**1010**	10	**1010**	10

Problema relacionado. Sumar 1111 y 1100.

Resta binaria

Las cuatro reglas básicas para la resta de números binarios son:

0 − 0 = 0	
1 − 1 = 0	
1 − 0 = 1	
10 − 1 = 1	0 − 1 con acarreo negativo de 1

Cuando se restan números, algunas veces se genera un acarreo negativo que pasa a la siguiente columna de la izquierda. En binario, sólo se produce acarreo negativo cuando se intenta restar 1 de 0. En este caso, cuando se acarrea un 1 a la siguiente columna de la izquierda, en la columna que se está restando se genera un 10, y entonces debe aplicarse la última de las cuatro reglas enumeradas. Los Ejemplos 2.8 y 2.9 ilustran la resta binaria, y se muestra también la resta decimal equivalente.

EJEMPLO 2.8 Realizar las siguientes sustracciones binarias: **(a)** 11 − 01 **(b)** 11 − 10.

Solución

(a)	11	3		**(b)**	11	3
	− 01	− 1			− 10	− 2
	10	2			**01**	1

En este ejemplo no se han generado acarreos negativos. El número binario 01 es el mismo que el 1.

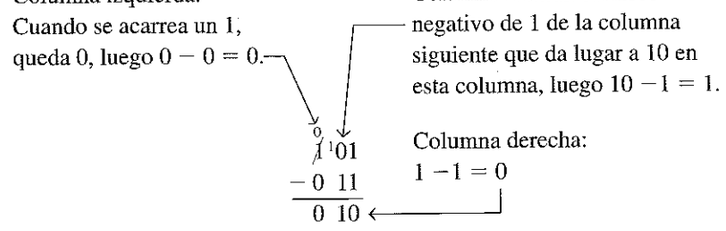
Problema relacionado. Restar 100 de 111.

EJEMPLO 2.9 Restar 011 de 101.

Solución

101	5
−011	−3
010	2

Examinemos detalladamente cómo se ha obtenido la resta de los dos números binarios, ya que es necesario un acarreo negativo. Empezamos por la columna de la derecha.



Problema relacionado. Restar 101 de 110.

Multiplicación binaria

Las cuatro reglas básicas de la multiplicación de bits son las siguientes:

$$0 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 0 = 0$$
$$1 \times 1 = 1$$

La multiplicación con números binarios se realiza de la misma forma que con números decimales. Se realizan los productos parciales, desplazando cada producto parcial una posición a la izquierda, y luego se suman dichos productos. El Ejemplo 2.10 ilustra este procedimiento, y muestra como referencia la multiplicación decimal equivalente.

EJEMPLO 2.10 Realizar las siguientes multiplicaciones binarias:
(a) 11 x 11 **(b)** 101 x 111

Solución

(a)

$$\begin{array}{rr} & 11 \\ & \times\ 11 \\ \text{Productos parciales} & 11 \\ & +\ 11 \\ \hline & \mathbf{1001} \end{array} \qquad \begin{array}{r} 3 \\ \times\ 3 \\ \hline 9 \end{array}$$

(b)

$$\begin{array}{rr} & 111 \\ & \times\ 101 \\ \text{Productos parciales} & 111 \\ & 000 \\ & +\ 111 \\ \hline & \mathbf{100011} \end{array} \qquad \begin{array}{r} 7 \\ \times\ 5 \\ \hline 35 \end{array}$$

Problema relacionado. Multiplicar 1101 x 1010.

División binaria

La división binaria sigue el mismo procedimiento que la división decimal, como muestra el Ejemplo 2.11. Obsérvese que también se facilitan los equivalentes decimales.

EJEMPLO 2.11 Realizar las siguientes divisiones binarias:
(a) 110 ÷ 11 **(b)** 110 ÷ 10

Solución

(a)

$$\begin{array}{r} \mathbf{10} \\ 11\overline{)110} \\ \underline{11} \\ 000 \end{array} \qquad \begin{array}{r} 2 \\ 3\overline{)6} \\ \underline{6} \\ 0 \end{array}$$

(b)

$$\begin{array}{r} \mathbf{11} \\ 10\overline{)110} \\ \underline{10} \\ 10 \\ \underline{10} \\ 00 \end{array} \qquad \begin{array}{r} 3 \\ 2\overline{)6} \\ \underline{6} \\ 0 \end{array}$$

Problema relacionado. Dividir 1100 entre 100.

REVISIÓN DE LA SECCIÓN 2.4

1. Realizar las siguientes sumas binarias:
   (a) 1101 + 1010 **(b)** 10111 + 01101
2. Realizar las siguientes sustracciones binarias:
   (a) 1101 – 0100 **(b)** 1001 – 0111
3. Realizar las operaciones binarias indicadas:
   (a) 110 x 111 **(b)** 1100 ÷ 011

2.5 ■ COMPLEMENTO A 1 Y COMPLEMENTO A 2 DE LOS NÚMEROS BINARIOS

El complemento a 1 y el complemento a 2 de un número binario son importantes porque permiten la representación de números negativos. La aritmética en complemento a 2 se usa comúnmente en las computadoras para manipular los números negativos.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Pasar un número binario a su formato en complemento a 1. ■ Pasar un número binario a su formato en complemento a 2 utilizando cualquiera de los dos posibles métodos.

Obtención del complemento a 1 de un número binario

El **complemento a 1** de un número binario se obtiene cambiando todos los 1s por 0s, y todos los 0s por 1s, como se ilustra a continuación:

$$\begin{array}{cccccccc} 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \end{array} \qquad \begin{array}{l} \text{Número binario} \\ \\ \text{Complemento a 1} \end{array}$$

Ejemplo de aplicación

La forma más sencilla de obtener el complemento a 1 de un número binario mediante un circuito digital es utilizar inversores en paralelo (circuitos NOT), como muestra la Figura 2.2 para un número binario de 8 bits.

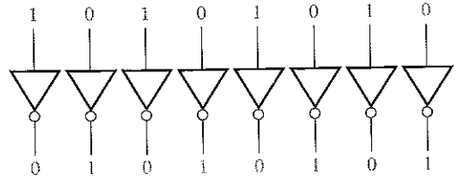


FIGURA 2.2
Ejemplo de inversores utilizados para obtener el complemento a 1 de un número binario.

Obtención del complemento a 2 de un número binario

El complemento a 2 de un número binario se obtiene sumando 1 al LSB del complemento a 1.

Complemento a 2 = (complemento a 1) + 1

El Ejemplo 2.12 muestra cómo se calcula el complemento a 2.

EJEMPLO 2.12 Hallar el complemento a 2 de 10110010

Solución

	10110010	Número binario
	01001101	Complemento a 1
+	1	Se suma 1
	01001110	Complemento a 2

Problema relacionado. Determinar el complemento a 2 de 11001011.

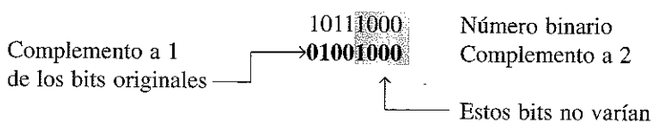
Un método alternativo para obtener el complemento a 2 de un número binario es el siguiente:

1. Se empieza por la derecha con el LSB y se escriben los bits como están hasta encontrar el primer 1, incluido éste.
2. Se calcula el complemento a 1 de los bits restantes.

El Ejemplo 2.13 ilustra estos pasos.

EJEMPLO 2.13 Calcular el complemento a 2 de 10111000, utilizando el método alternativo.

Solución



Problema relacionado. Calcular el complemento a 2 de 11000000.

Ejemplo de aplicación

El complemento a 2 de un número binario negativo puede calcularse utilizando inversores y un sumador, como se muestra en la Figura 2.3. Ésta ilustra cómo un número de bits puede convertirse en su complemento a 2, invirtiendo en primer lugar

cada bit (haciendo el complemento a 1) y luego, con el sumador, sumando 1 al complemento a 1.

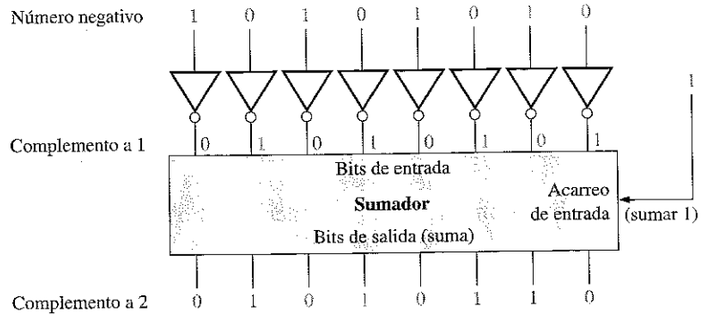


FIGURA 2.3
Ejemplo de obtención del complemento a 2 de un número binario negativo.

Para convertir un número en complemento a 1 o en complemento a 2 al formato binario real (no complementado) se usan los dos mismos procedimientos que acabamos de describir. Para convertir el complemento a 1 al binario real, se invierten todos los bits. Para convertir el complemento a 2 al binario real, primero se obtiene el complemento a 1 y se suma 1 al resultado obtenido.

REVISIÓN DE LA SECCIÓN 2.5

1. Determinar el complemento a 1 de los siguientes números binarios:
   (a) 00011010 **(b)** 11110111 **(c)** 10001101
2. Determinar el complemento a 2 de los siguientes números binarios:
   (a) 00010110 **(b)** 11111100 **(c)** 10010001

2.6 ■ NÚMEROS CON SIGNO

Los sistemas digitales, tales como la computadora, deben ser capaces de manejar números positivos y negativos. Un número binario con signo queda determinado por su magnitud y su signo. El signo indica si un número es positivo o negativo, y la magnitud es el valor del número. Existen tres formatos binarios para representar los números enteros con signo: signo-magnitud, complemento a 1 y complemento a 2. Los números no enteros y muy grandes o muy pequeños pueden expresarse en formato de coma flotante.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Expresar los números positivos y negativos en formato signo-magnitud. ■ Expresar los números positivos y negativos en complemento a 1. ■ Expresar los números positivos y negativos en complemento a 2. ■ Determinar el valor decimal de los números binarios con signo. ■ Expresar un número binario en formato de coma flotante.

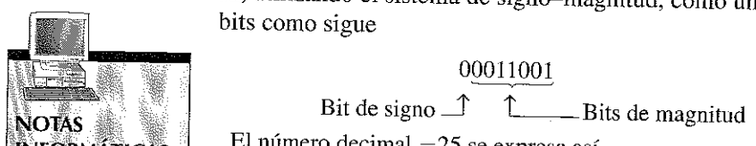
El bit de signo

El bit más a la izquierda en un número binario con signo es el **bit de signo**, que indica si el número es positivo o negativo.

Se utiliza un 0 para el signo positivo y un 1 para el signo negativo.

Sistema signo-magnitud

Cuando un número binario con signo se representa en formato signo-magnitud, el bit más a la izquierda es el bit de signo y los bits restantes son los bits de magnitud. Los bits de magnitud son el número binario real (no complementado) tanto para los números positivos como para los negativos. Por ejemplo, el número decimal +25 se expresa, utilizando el sistema de signo–magnitud, como un número binario con signo de 8 bits como sigue



El número decimal −25 se expresa así

10011001

Obsérvese que la única diferencia entre +25 y –25 es el bit de signo, ya que los bits de magnitud representan el binario real tanto para números positivos como negativos.

En el sistema signo-magnitud, un número negativo tiene los mismos bits de magnitud que el correspondiente número positivo, pero el bit de signo es un 1 en lugar de un cero.

Sistema del complemento a 1

Los números positivos en el sistema del complemento a 1 se representan de la misma forma que los números positivos en el formato signo-magnitud. Sin embargo, los números negativos son el complemento a 1 del correspondiente número positivo. Por ejemplo, usando ocho bits, el número decimal –25 se expresa como el complemento a 1 de +25 (00011001), es decir,

11100110

En el sistema del complemento a 1, un número negativo es el complemento a 1 del correspondiente número positivo.

Sistema del complemento a 2

Los números positivos en el sistema del complemento a 2 se representan de la misma forma que en los sistemas de complemento a 1 y de signo-magnitud. Los números negativos son el complemento a 2 del correspondiente número positivo. De nuevo, usando ocho bits, tomamos –25 y lo expresamos como el complemento a 2 de +25 (00011001), es decir,

11100111

NOTAS INFORMÁTICAS Las computadoras utilizan el complemento a 2 para los números enteros negativos en todas las operaciones aritméticas. La razón de ello es que la sustracción de un número es lo mismo que sumar el complemento a 2 del número. Las computadoras obtienen el complemento a 2 invirtiendo los bits y sumando 1, utilizando instrucciones especiales que generan el mismo resultado que el sumador visto en el ejemplo de aplicación anterior.

En el sistema del complemento a 2, un número negativo es el complemento a 2 del correspondiente número positivo.

EJEMPLO 2.14 Expresar el número decimal –39 como un número de 8 bits en los sistemas signo-magnitud, complemento a 1 y complemento a 2.

Solución. En primer lugar, escribimos el número binario de 8 bits para +39:

00100111

En el *sistema signo-magnitud*, –39 se obtiene cambiando el bit de signo a 1, y dejando los bits de magnitud como están. El número es:

10100111

En el *sistema de complemento* a 1, –39 se obtiene calculando el complemento a 1 de +39 (00100111):

11011000

En el *sistema de complemento* a 2, –39 se obtiene calculando el complemento a 2 de +39 (00100111) del siguiente modo:

```
  11011000    Complemento a 1
+        1
----------
  11011001    Complemento a 2
```

Problema relacionado. Expresar +19 y –19 en los formatos signo-magnitud, complemento a 1 y complemento a 2, con 8 bits.

El valor decimal de los números con signo

Signo-magnitud. Los valores decimales de los números positivos y negativos en el sistema de signo-magnitud se determinan sumando los pesos de todas las posiciones de los bits de magnitud, cuando son 1s, e ignorando aquellas posiciones en las que haya ceros. El signo se determina por medio del examen del bit de signo. El Ejemplo 2.15 ilustra este proceso.

EJEMPLO 2.15 Determinar el valor decimal del número binario con signo expresado como signo-magnitud: 10010101.

Solución

Los siete bits de magnitud y sus pesos potencias de dos son los siguientes:

2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	1	0	1	0	1

Sumando los pesos de las posiciones donde hay 1s, tenemos

$$16 + 4 + 1 = 21$$

El bit de signo es 1, por tanto, el número es **–21**.

Problema relacionado. Determinar el valor decimal del número en formato signo-magnitud 01110111.

Complemento a 1. Los valores decimales de los números positivos en el sistema de complemento a 1 se determinan sumando los pesos de todas las posiciones de bit donde haya 1s, y se ignoran aquellas posiciones donde haya ceros. Los valores decimales de los números negativos se determinan asignando el valor negativo al peso del bit de signo, y sumando todos los pesos donde haya 1s, y añadiendo luego 1 al resultado. El Ejemplo 2.16 ilustra esto.

EJEMPLO 2.16 Determinar el valor decimal de los números binarios con signo expresados en complemento a 1:
(a) 00010111 **(b)** 11101000.

Solución

(a) Los bits y sus pesos según las potencias de dos para el número positivo son:

$$\begin{array}{cccccccc} -2^7 & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{array}$$

Sumando los pesos donde hay 1s.

$$16 + 4 + 2 + 1 = +\mathbf{23}$$

(b) Los bits y sus pesos según las potencias de dos para el número negativo son los siguientes (obsérvese que el bit de signo negativo tiene un peso de -2^7, es decir, -128):

$$\begin{array}{cccccccc} -2^7 & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{array}$$

Sumando los pesos donde hay 1s.

$$-128 + 64 + 32 + 8 = -24$$

Sumando 1 al resultado, el número final es:

$$-24 + 1 = -\mathbf{23}$$

Problema relacionado. Determinar el valor decimal del número en complemento a 1: 11101011.

Complemento a 2. Los valores decimales de los números positivos y negativos en el sistema de complemento a 2 se determinan sumando los pesos de todas las posiciones de bit donde haya 1s e ignorando aquellas posiciones donde haya ceros. El peso del bit de signo en un número negativo viene determinado por su valor negativo. El Ejemplo 2.17 ilustra esto.

EJEMPLO 2.17 Determinar los valores decimales de los números binarios con signo expresados en complemento a 2:

(a) 01010110 **(b)** 10101010.

Solución

(a) Los bits y sus pesos según las potencias de dos para el número positivo son los siguientes:

$$\begin{array}{cccccccc} -2^7 & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{array}$$

Sumando los pesos donde hay 1s.

$$64 + 16 + 4 + 2 = +\mathbf{86}$$

(b) Los bits y sus pesos según las potencias de dos para el número negativo son los siguientes (obsérvese que el bit de signo negativo tiene un peso de -2^7, es decir, -128):

$$\begin{array}{cccccccc} -2^7 & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{array}$$

Sumando los pesos donde hay 1s.

$$-128 + 32 + 8 + 2 = -\mathbf{86}$$

Problema relacionado. Determinar el valor decimal del número en complemento a 2: 11010111.

A partir de estos ejemplos, se puede deducir una de las razones por las que se prefiere el sistema de complemento a 2 para representar los números con signo: simplemente, es que requiere una adición de pesos independientemente de que el número sea positivo o negativo. El sistema signo-magnitud requiere dos pasos: sumar los pesos de los bits de magnitud y examinar el bit de signo para determinar si el número es positivo o negativo. El sistema de complemento a 1 requiere añadir 1 a la suma de los pesos para los números negativos, pero no para los números positivos. También, generalmente, el sistema de complemento a 1 no se utiliza porque son posibles dos representaciones del cero (00000000 o 11111111).

Rango de representación de los números con signo

Para los ejemplos hemos utilizado números de 8 bits, puesto que la agrupación de 8 bits es un estándar en la mayoría de las computadoras, y recibe el nombre especial de **byte**. Utilizando 8 bits, se pueden representar 256 números diferentes. Combinando dos bytes se obtienen dieciséis bits, y se pueden representar 65.536 números diferentes. Combinando cuatro bytes se obtienen 32 bits, y se pueden representar $4{,}295 \times 10^9$ números diferentes, y así sucesivamente. La fórmula para calcular el número de combinaciones diferentes de n bits es:

$$\text{N° total de combinaciones} = 2^n$$

Para los números con signo en complemento a 2, el rango de valores para números de n bits es:

$$-(2^{n-1}) \text{ a } +(2^{n-1}-1)$$

habiendo en cada caso un bit de signo y n–1 bits de magnitud. Por ejemplo, con cuatro bits pueden representarse números en complemento a 2 en el rango de $-(2^3) = -8$ hasta $2^3 - 1 = +7$. Del mismo modo, con ocho bits, se pueden abarcar desde –128 hasta +127; con dieciséis bits se puede ir desde –32.768 hasta +32.767, y así sucesivamente.

Números en coma flotante

Para representar números **enteros** muy grandes, son necesarios muchos bits. También se produce un problema cuando se necesita representar números con parte entera y parte fraccionaria, tal como 23,5618. El sistema de numeración en coma flotante, basado en la notación científica, permite representar números muy grandes y números muy pequeños sin aumentar el número de bits, y también sirve para representar números con parte fraccionaria y parte entera.

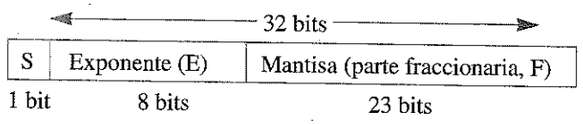
Un **número en coma flotante** (también conocido como *número real*) tiene dos partes más un signo. La **mantisa** es la parte del número en coma flotante que representa la magnitud del número. El **exponente** es la parte de un número en coma flotante que representa el número de lugares que se va a desplazar el punto decimal (o punto binario).

Un ejemplo de número decimal nos será útil para comprender el concepto básico de los números en coma flotante. Consideremos un número decimal que, en formato entero, es 241.506.800. La mantisa es, 2415068 y el exponente es 9. Cuando el entero se expresa como un número en coma flotante, se normaliza desplazando el punto decimal a la izquierda de todos los dígitos, de modo que la mantisa es un número fraccionario y el exponente es una potencia de 10. Este número en coma flotante se escribe:

$$0{,}2415068 \times 10^9$$

Para los números en coma flotante binarios, el formato definido por el estándar 754-1985 ANSI/IEEE puede tomar tres formas: *simple precisión, doble precisión* y *precisión ampliada*. Todos ellos utilizan el mismo formato básico excepto por el número de bits. Los números en coma flotante de simple precisión tienen 32 bits, los de doble precisión tienen 64 bits y los de precisión ampliada tienen 80 bits. Vamos a restringir nuestra exposición al formato de los números en coma flotante de simple precisión.

Números binarios en coma flotante de simple precisión. En el formato estándar para un número binario de simple precisión, el bit de signo (S) es el que se encuentra más a la izquierda, el exponente (E) incluye los siguientes 8 bits y la mantisa o parte fraccionaria (F) incluye los restantes 23 bits, como se muestra a continuación.



En la mantisa o parte fraccionaria, se entiende que el punto binario estará a la izquierda de los 23 bits. Realmente, la mantisa consta de 24 bits, ya que, en cualquier número binario, el bit más a la izquierda (más significativo) es siempre un 1. Por tanto, este 1 se entiende que estará allí aunque no ocupe una posición de bit real.

Los 8 bits de los que consta el exponente representan un *exponente desplazado* que se ha obtenido mediante la adición de 127 al exponente real. El propósito de este desplazamiento es poder definir números muy grandes o muy pequeños sin necesidad de emplear un bit de signo diferente para el exponente. El exponente desplazado permite emplear un rango de valores para los exponentes comprendido entre –126 y +128.

Para ilustrar cómo un número binario se expresa en formato de coma flotante, utilizaremos como ejemplo el número binario 1011010010001. En primer lugar, podemos expresarlo como 1 más un número binario fraccionario, desplazando la coma binaria doce posiciones a la izquierda y multiplicándolo después por la apropiada potencia de 2.

$$1011010010001 = 1{,}011010010001 \times 2^{12}$$

Suponiendo que se trate de un número positivo, el bit de signo (S) es 0. El exponente, 12, se expresa como un exponente desplazado añadiéndole 127 (12+127=139). El exponente desplazado (E) se expresa como el número binario 10001011. La mantisa es la parte fraccionaria (F) del número binario, ,011010010001. Dado que siempre existe un 1 a la izquierda de la coma binaria en la expresión de la potencia de dos, no se incluye en la mantisa. El número en coma flotante completo es

S	E	F
0	10001011	01101001000100000000000

A continuación, veamos cómo evaluar un número binario que ya está en formato de coma flotante. El método general para determinar el valor de un número en coma flotante se expresa mediante la siguiente fórmula:

$$\text{Número} = (-1)^S(1 + F)(2^{E-127})$$

Para ilustrar este método, consideremos el siguiente número binario en coma flotante:

1	10010001	10001110001000000000000

El bit de signo es 1. El exponente desplazado es 10010001 = 145. Aplicando la fórmula, obtenemos

$$\begin{aligned}\text{Número} &= (-1)^1(1.10001110001)(2^{145-127}) \\ &= (-1)(1.10001110001)(2^{18}) = -110001110001000000\end{aligned}$$

Este número binario en coma flotante es equivalente a –407.688 en decimal. Dado que el exponente puede ser cualquier número comprendido entre –126 y +128, pueden expresarse números extremadamente grandes y pequeños. Un número en coma flotante de 32 bits puede reemplazar a un número entero binario utilizando 129 bits. Dado que el exponente determina la posición de la coma binaria, se pueden representar números que contengan tanto parte entera como parte fraccionaria.

Existen dos excepciones para el formato de los números en coma flotante: el número 0,0 se representa utilizando sólo ceros, e infinito se representa utilizando sólo unos en el exponente y ceros en la mantisa.

EJEMPLO 2.18 Convertir el número decimal $3{,}248 \times 10^4$ en un número binario en formato de coma flotante de simple precisión.

Solución

Conversión del número decimal a binario.

$$3{,}248 \times 10^4 = 32480 = 111111011100000_2 = 1.11111011100000 \times 2^{14}$$

El bit más significativo (MSB) no ocupará una posición de bit, dado que siempre es 1. Por tanto, la mantisa es el número fraccionario binario de 23 bits 11111011100000000000000 y el exponente desplazado es

$$14 + 127 = 141 = 10001101_2$$

El número en coma flotante completo es

0	10001101	11111011100000000000000

Problema relacionado. Determinar el valor binario del siguiente número binario en coma flotante:

0 10011000 10000100010100110000000

REVISIÓN DE LA SECCIÓN 2.6

1. Expresar el número decimal +9 como un número binario de 8 bits en formato signo–magnitud.
2. Expresar el número decimal –33 como un número binario de 8 bits en el sistema de complemento a 1.
3. Expresar el número decimal –46 como un número binario de 8 bits en el sistema de complemento a 2.
4. Enumerar las dos partes de un número con signo en formato de coma flotante.

2.7 ■ OPERACIONES ARITMÉTICAS DE NÚMEROS CON SIGNO

En la sección anterior, hemos aprendido cómo se representan, en tres sistemas diferentes, los números con signo. En esta sección, aprenderemos cómo se suman, restan, multiplican y dividen estos números. Puesto que el complemento a 2 es el sistema de representación de números con signo más ampliamente utilizado en las computadoras y en los sistemas basados en microprocesador, esta sección se limita a cubrir la aritmética en complemento a 2. Los procedimientos que se cubren pueden extenderse a los demás sistemas, si fuera necesario.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Sumar números binarios con signo. ■ Explicar cómo las computadoras suman cadenas de números. ■ Definir *desbordamiento* (overflow). ■ Restar números binarios con signo. ■ Multiplicar números binarios con signo utilizando el método directo de suma. ■ Multiplicar números binarios con signo utilizando el método de productos parciales. ■ Dividir números binarios con signo.

NOTAS INFORMÁTICAS Además del procesador o unidad central de procesamiento (UCP o CPU), las computadoras utilizan *coprocesadores* para realizar cálculos matemáticos complejos utilizando números en coma flotante. El propósito es incrementar el rendimiento liberando al procesador para que lleve a cabo otras tareas. El coprocesador matemático también se conoce como la unidad de coma flotante (FPU, floating-point unit).

Suma

Los dos números en una suma se denominan **sumandos**. El resultado es la **suma**. Cuando se suman dos números binarios con signo pueden producirse cuatro casos:

1. Ambos números son positivos.
2. El número positivo es mayor que el negativo en valor absoluto.
3. El número negativo es mayor que el positivo en valor absoluto.
4. Ambos números son negativos.

Veamos caso por caso utilizando números con signo de 8 bits como ejemplos. Como referencia se presentan los números decimales equivalentes.

Ambos números son positivos:

```
  00000111      7
+ 00000100    + 4
  00001011     11
```

La suma es positiva y, por tanto, es un número binario real (no complementado).

El número positivo es mayor que el número negativo en valor absoluto:

```
                  00001111     15
                + 11111010   + −6
                  00001001      9
Acarreo que se descarta ⟶ 1
```

El bit de acarreo final no se tiene en cuenta. La suma es positiva y, por tanto, es un número binario real (no complementado)

El número negativo es mayor que el número positivo en valor absoluto:

```
  00010000     16
+ 11101000   + −24
  11111000     −8
```

La suma es negativa y, por tanto, está en complemento a 2.

Ambos números son negativos:

```
                  11111011     −5
                + 11110111   + −9
                  11110010    −14
Acarreo que se descarta ⟶ 1
```

El bit de acarreo final no se tiene en cuenta. La suma es negativa y, por tanto, está en complemento a 2.

En una computadora, los números negativos se almacenan en formato complemento a 2, por lo que, como puede verse, el procedimiento de suma es muy sencillo: *sumar los dos números y descartar cualquier bit de acarreo final.*

Condición de desbordamiento (overflow). Cuando se suman dos números y el número de bits requerido para representar la suma excede al número de bits de los dos números, se produce un **desbordamiento**, que se indica mediante un bit de signo incorrecto. Un desbordamiento se puede producir sólo cuando ambos números son positivos o negativos. El siguiente ejemplo con números de 8 bits ilustrará esta condición.

```
  01111101      125
+ 00111010    +  58
  10110111      183
```

Signo incorrecto
Magnitud incorrecta

En este ejemplo, la suma, 183 requiere ocho bits de magnitud. Puesto que los números tienen siete bits de magnitud (un bit es el bit de signo), se produce un acarreo en el bit de signo que da lugar a la indicación de desbordamiento.

Suma de números de dos en dos. Ahora vamos a ver la suma de una cadena de números, sumando de dos en dos. Esto se puede conseguir sumando los dos primeros números, luego se suma el tercer número a la suma de los dos primeros, después se suma el cuarto número al resultado anterior, y así sucesivamente. Así es como las computadoras suman cadenas de números. La suma de números dos a dos se ilustra en el Ejemplo 2.19.

EJEMPLO 2.19 Sumar los números con signo: 01000100, 00011011, 00001110 y 00010010.

Solución. Como referencia se proporciona el equivalente decimal.

```
  68      01000100    Suma de los dos primeros números
+ 27    + 00011011
  95      01011111    Primera suma
+ 14    + 00001110    Suma del tercer número
 109      01101101    Segunda suma
+ 18    + 00010010    Suma del cuarto número
 127      01111111    Suma final.
```

Problema relacionado. Sumar 00110011, 10111111 y 01100011. Son números con signo.

Sustracción

La sustracción es un caso especial de la suma. Por ejemplo, restar +6 (el **sustraendo**) de +9 (el **minuendo**) es equivalente a sumar −6 a +9. Básicamente, *la operación de la sustracción cambia el signo del sustraendo y le suma al minuendo.* El resultado de una sustracción se denomina **diferencia**.

El signo de un número binario positivo o negativo se cambia calculando su complemento a 2.

Por ejemplo, el complemento a 2 del número positivo 00000100 (+4) es 11111100, que es −4, tal como demuestra la siguiente evaluación de suma de pesos:

$$-128 + 64 + 32 + 16 + 8 + 4 = -4$$

Veamos otro ejemplo, el complemento a 2 del número negativo 11101101 (−19) es 00010011, que es +19 como demuestra la siguiente evaluación:

$$16 + 2 + 1 = 19$$

Puesto que la sustracción es sencillamente una suma con el signo del sustraendo cambiado, el procedimiento se establece como sigue:

Para restar dos números con signo se calcula el complemento a 2 del sustraendo y se suman descartando cualquier bit de acarreo final.

El Ejemplo 2.20 ilustra el procedimiento de la sustracción.

EJEMPLO 2.20 Realizar las siguientes sustracciones de números con signo:

(a) 00001000 − 00000011 **(b)** 00001100 − 11110111
(c) 11100111 − 00010011 **(d)** 10001000 − 11100010

Solución. Como en otros ejemplos, se facilitan los equivalentes decimales como referencia.

(a) En este caso, 8 − 3 = 8 + (−3) = 5

```
                00001000   Minuendo (+8)
              + 11111101   Complemento a 2 del sustraendo (−3)
Acarreo que se descarta ⟶ 1 00000101   Diferencia (+5)
```

(b) En este caso, 12 − (−9) = 12 + 9 = 21

```
  00001100   Minuendo (+12)
+ 00001001   Complemento a 2 del sustraendo (+9)
  00010101   Diferencia (+21)
```

(c) En este caso, −25 − (+19) = −25 + (−19) = −44

```
                11100111   Minuendo (−25)
              + 11101101   Complemento a 2 del sustraendo (−19)
Acarreo que se descarta ⟶ 1 11010100   Diferencia (−44)
```

(d) En este caso, −120 − (−30) = −120 + 30 = −90

```
  10001000   Minuendo (−120)
+ 00011110   Complemento a 2 del sustraendo (+30)
  10100110   Diferencia (−90)
```

Problema relacionado. Restar 01000111 de 01011000.

Multiplicación

Los números en una multiplicación se denominan **multiplicando, multiplicador** y **producto**. La siguiente multiplicación decimal ilustra estos términos:

```
   8    Multiplicando
 × 3    Multiplicador
  24    Producto
```

La operación de la multiplicación en muchas computadoras se realiza utilizando la suma. Como ya se ha visto, la sustracción se hace como una suma; ahora vamos a ver cómo se hace la multiplicación.

La *suma directa* y los *productos parciales* son dos métodos básicos para realizar la multiplicación utilizando la suma. En el método de la suma directa, se suma el multiplicando un número de veces igual al multiplicador. En el ejemplo decimal anterior (3 x 8), se suma tres veces el multiplicando: 8 + 8 + 8 = 24. La desventaja de este método es que será muy largo cuando el multiplicador sea un número grande. Si, por ejemplo, se multiplica 75 x 350, se debe sumar 75 veces el número 350.

El método de los productos parciales es quizá el más común, ya que es la forma de multiplicar manualmente. El multiplicando se multiplica por cada dígito del multiplicador, empezando por el dígito menos significativo. El resultado de la multiplicación del multiplicando por un dígito del multiplicador se denomina *producto parcial*. Cada producto parcial se desplaza una posición a la izquierda y, cuando se han obtenido todos los productos parciales, se suman para obtener el producto final. Aquí tiene un ejemplo con números decimales.

```
   239    Multiplicando
 ×123     Multiplicador
   717    Primer producto parcial (3 × 239)
  478     Segundo producto parcial (2 × 239)
+ 239     Tercer producto parcial (1 × 239)
 29.397   Producto final
```

El signo del producto de una multiplicación depende de los signos del multiplicando y del multiplicador, de acuerdo con las siguientes reglas:

- **Si son del mismo signo, el producto es positivo.**
- **Si son de diferente signo, el producto es negativo.**

Cuando dos números binarios se multiplican, ambos números deben estar en formato binario real (no complementado). El método de la suma directa se ilustra en el Ejemplo 2.21 sumando los números binarios de dos en dos.

EJEMPLO 2.21 Multiplicar los números binarios con signo siguientes: 01001101 (multiplicando) y 00000100 (multiplicador), utilizando el método de suma directa.

Solución

Puesto que los dos números son positivos, y ambos están en formato real, el producto será positivo. El valor decimal del multiplicador es 4, luego hay que sumar cuatro veces el multiplicando del siguiente modo:


```
  01001101    Primera vez
+ 01001101    Segunda vez
  10011010    Suma parcial
+ 01001101    Tercera vez
  11100111    Suma parcial
+ 01001101    Cuarta vez
 100110100    Producto
```

Puesto que el bit de signo del multiplicando es 0, no tiene ningún efecto sobre el resultado. Todos los bits del producto son bits de magnitud.

Problema relacionado. Multiplicar 01100001 por 00000110.

Ahora vamos a ver el método de los productos parciales para la multiplicación binaria. Los pasos básicos del procedimiento son los siguientes:

Paso 1. Determinar si los signos del multiplicando y del multiplicador son iguales o diferentes. Así se determina el signo que tendrá el producto.

Paso 2. Poner cualquier número negativo en formato real (no complementado). Puesto que la mayoría de las computadoras almacenan los números negativos en complemento a 2, se requiere la operación de complemento a 2 para obtener el número negativo en formato real.

Paso 3. Empezar por el bit del multiplicador menos significativo y generar los productos parciales. Cuando el bit del multiplicador es 1, el producto parcial es igual al multiplicando. Cuando el bit del multiplicador es 0, el producto parcial es cero. Cada sucesivo producto parcial debe desplazarse un bit a la izquierda.

Paso 4. Sumar cada producto parcial a la suma de los productos parciales anteriores para obtener el producto final.

Paso 5. Si el bit de signo que se había determinado en el paso 1 es negativo, calcular el complemento a 2 del producto. Si es positivo, dejar el producto en formato real. Añadir el bit de signo al producto.

El Ejemplo 2.22 ilustra estos pasos.

EJEMPLO 2.22 Multiplicar los números binarios con signo: 01010011 (multiplicando) y 11000101 (multiplicador).

Solución

Paso 1. El bit de signo del multiplicando es 0 y el bit de signo del multiplicador es 1. El bit de signo del producto será 1 (negativo).

Paso 2. Calcular el complemento a 2 del multiplicador para ponerlo en formato real

$$11000101 \longrightarrow 00111011$$

Pasos 3 y 4. El procedimiento de multiplicación es como sigue. Obsérvese que en estos pasos sólo se usan los bits de magnitud.

1010011	Multiplicando
× 0111011	Multiplicador
1010011	Primer producto parcial
+ 1010011	Segundo producto parcial
11111001	Suma del primero y el segundo
+ 0000000	Tercer producto parcial
011111001	Suma
+ 1010011	Cuarto producto parcial
1110010001	Suma
+ 1010011	Quinto producto parcial
100011000001	Suma
+ 1010011	Sexto producto parcial
1001100100001	Suma
+ 0000000	Séptimo producto parcial
1001100100001	Producto final

Paso 5. Puesto que el signo del producto es 1, como se determinó en el paso 1, calcular el complemento a 2 del producto.

$$1001100100001 \longrightarrow 0110011011111$$

Se añade el bit de signo $\longrightarrow$ **1 0110011011111**

Problema relacionado. Verificar que la multiplicación es correcta convirtiendo los números binarios a decimales y realizando la multiplicación.

División

Los números en una división son el **dividendo**, el **divisor** y el **cociente**. Éstos se ilustran en el siguiente formato estándar de división:

$$\frac{\text{dividendo}}{\text{divisor}} = \text{cociente}$$

En las computadoras, la operación de la división se lleva a cabo utilizando la sustracción. Puesto que la sustracción se hace con un sumador, la división también puede realizarse con un sumador.

El resultado de una división es el cociente: el cociente es el número de veces que el divisor estará contenido en el dividendo. Esto significa que el divisor puede restarse del dividendo un número de veces igual al cociente, tal como ilustra la división de 21 entre 7:

21	Dividendo
− 7	Primera sustracción del divisor
14	Primer resto parcial
− 7	Segunda sustracción del divisor
7	Segundo resto parcial
− 7	Tercera sustracción del divisor
0	Resto cero

En este sencillo ejemplo, el divisor se ha restado del dividendo tres veces antes de obtener resto cero. Por tanto, el cociente es 3.

El signo del cociente depende de los signos del dividendo y del divisor, de acuerdo con las dos reglas siguientes.

- **Si son del mismo signo, el cociente es positivo.**
- **Si son de diferente signo, el cociente es negativo.**

Cuando se dividen dos números binarios, ambos números deben estar en formato real (no complementado). Los pasos básicos en un procedimiento de división son los siguientes:

Paso 1. Determinar si los signos del dividendo y del divisor son iguales o diferentes. Esto determina qué signo tendrá el cociente. Inicializar el cociente a cero.

Paso 2. Restar el divisor del dividendo utilizando la suma en complemento a 2, para obtener el primer resto parcial, y sumar 1 al cociente. Si este resto parcial es positivo, ir al paso 3. Si el resto parcial es cero o negativo, la división se ha terminado.

Paso 3. Restar el divisor del resto parcial y sumar 1 al cociente. Si el resultado es positivo, repetir para el siguiente resto parcial. Si el resultado es cero o negativo, la división se ha terminado.

Continuar restando el divisor del dividendo y los restos parciales hasta que el resultado sea cero o negativo. Contar el número de veces que se ha restado el divisor y se obtendrá el cociente. El Ejemplo 2.23 ilustra estos pasos utilizando números binarios con signo de 8 bits.

EJEMPLO 2.23 Dividir 01100100 entre 00011001.

Solución

Paso 1. El signo de ambos números es positivo, luego el cociente será positivo. Inicialmente, el cociente es cero: 00000000.

Paso 2. Restar el divisor del dividendo utilizando la suma en complemento a 2 (recuerde que los acarreos finales no se tienen en cuenta):

01100100	Dividendo
+11100111	Complemento a 2 del divisor
01001011	Primer resto parcial positivo

Sumar 1 al cociente: 00000000 + 00000001 = 00000001.

Paso 3. Restar el divisor del primer resto parcial utilizando la suma en complemento a 2.

01001011	Primer resto parcial
+11100111	Complemento a 2 del divisor
00110010	Segundo resto parcial positivo

Sumar 1 al cociente: 00000001 + 00000001 = 00000010.

Paso 4. Restar el divisor del segundo resto parcial utilizando la suma en complemento a 2.

```
 00110010    Segundo resto parcial
+11100111    Complemento a 2 del divisor
 00011001    Tercer resto parcial positivo
```

Sumar 1 al cociente: 00000010 + 00000001 = 00000011.

Paso 5. Restar el divisor del tercer resto parcial utilizando la suma en complemento a 2.

```
 00011001    Tercer resto parcial
+11100111    Complemento a 2 del divisor
 00000000    Resto cero
```

Sumar 1 al cociente: 00000011 + 00000001 = **00000100** (cociente final). El proceso se ha completado.

Problema relacionado. Verificar que el procedimiento es correcto convirtiendo los números binarios a decimales y realizando la división.

REVISIÓN DE LA SECCIÓN 2.7

1. Enumerar los cuatro casos de suma de números.
2. Sumar 00100001 y 10111100.
3. Restar 00110010 de 01110111.
4. ¿Cuál es el signo del producto cuando se multiplican dos números negativos?
5. Multiplicar 01111111 por 00000101.
6. ¿Cuál es el signo del cociente cuando se divide un número positivo entre un número negativo?
7. Dividir 00110000 entre 00001100.

2.8 ■ NÚMEROS HEXADECIMALES

El sistema de numeración hexadecimal consta de dieciséis dígitos y se usan fundamentalmente como una forma simplificada de representar o escribir los números binarios, ya que es muy fácil la conversión entre binario y hexadecimal. Como probablemente se habrá podido comprobar, los números binarios largos son difíciles de leer y escribir, ya que es fácil omitir o transponer un bit. Puesto que las computadoras y microprocesadores sólo entienden los 1s y los 0s, es necesario emplear estos dígitos cuando se programa en "lenguaje máquina". Imaginemos tener que escribir una instrucción de sesenta bits para un sistema de microprocesador utilizando 1s y 0s. Es mucho más efectivo utilizar los números hexadecimales u octales. Los números octales se cubren en la Sección 2.9. El sistema hexadecimal se usa frecuentemente en computadoras y aplicaciones de microprocesadores.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Enumerar los dígitos hexadecimales. ■ Contar en hexadecimal. ■ Convertir de binario a hexadecimal. ■ Convertir de hexadecimal a binario. ■ Convertir de hexadecimal a decimal. ■ Convertir de decimal a hexadecimal. ■ Sumar números hexadecimales. ■ Restar números hexadecimales.

El sistema **hexadecimal** es un sistema en base dieciséis, es decir, está formado por 16 dígitos y **caracteres alfabéticos**. La mayoría de los sistemas digitales procesan grupos de datos binarios que son múltiplos de cuatro bits, lo que hace al número hexadecimal muy adecuado, ya que cada dígito hexadecimal se representa mediante un número binario de 4 bits, como se puede ver en la Tabla 2.3.

TABLA 2.3

Decimal	Binario	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

NOTAS INFORMÁTICAS
Con memorias de computadora en el rango de los gigabytes, especificar una dirección de memoria en binario es bastante complicado. Por ejemplo, hay que emplear 32 bits para especificar una dirección de una memoria de 4 gigabytes. Es mucho más sencillo expresar un código de 32 bits utilizando ocho dígitos hexadecimales.

Diez dígitos **numéricos** y seis caracteres alfabéticos forman el sistema de numeración hexadecimal. El uso de las letras A, B, C, D, E y F para representar números puede parecer extraño al principio, pero hay que tener en mente que cualquier sistema de numeración es sólo un conjunto de símbolos secuenciales. Si comprendemos qué cantidades representan estos símbolos, entonces la forma de los símbolos en sí tiene poca importancia, una vez que nos hayamos acostumbrado a utilizarlos. Utilizaremos el subíndice 16 para designar a los números hexadecimales y evitar así cualquier confusión con los números decimales. En ocasiones, puede verse la letra "h" detrás de un número hexadecimal.

Contar en hexadecimal

¿Cómo se cuenta en hexadecimal una vez que se ha llegado a la F? Sencillamente se inicia otra columna y se continúa así:

10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 2A, 2B, 2C, 2D, 2E, 2F, 30, 31...

Con dos dígitos hexadecimales, se puede contar hasta FF_{16}, que corresponde al decimal 255. Para continuar contando, son necesarios tres dígitos hexadecimales. Por ejemplo, 100_{16} es el 256 decimal, 101_{16} es el decimal 257, y así sucesivamente. El

máximo número hexadecimal de 3 dígitos es FFF_{16}, es decir el decimal 4095. El máximo número hexadecimal de 4 dígitos es $FFFF_{16}$, que es el decimal 65.535.

Conversión binario-hexadecimal

La conversión de un número binario en hexadecimal es un procedimiento muy sencillo. Simplemente se parte el número binario en grupos de 4 bits, comenzando por el bit más a la derecha, y se reemplaza cada grupo de 4 bits por su símbolo hexadecimal equivalente, como se ilustra en el Ejemplo 2.24.

EJEMPLO 2.24 Convertir a hexadecimal los siguientes números binarios:
(a) 1100101001010111 **(b)** 111111000101101001

Solución

(a) 1100 1010 0101 0111 → C A 5 7 = **$CA57_{16}$**

(b) 0011 1111 0001 0110 1001 → 3 F 1 6 9 = **$3F169_{16}$**

En el apartado (b) se han añadido dos ceros para completar el grupo de 4 bits de la izquierda.

Problema relacionado. Convertir a hexadecimal el número binario 1001111011110011100.

Conversión hexadecimal-binario

Para convertir un número hexadecimal en un número binario se realiza el proceso inverso, reemplazando cada símbolo hexadecimal por el grupo de cuatro bits adecuado, como ilustra el Ejemplo 2.25.

EJEMPLO 2.25 Determinar los números binarios correspondientes a los siguientes números hexadecimales:
(a) $10A4_{16}$ **(b)** $CF8E_{16}$ **(c)** 9742_{16}

Solución

(a) 1 0 A 4 → **1000010100100**

(b) C F 8 E → **1100111110001110**

(c) 9 7 4 2 → **1001011101000010**

En el apartado (a), el MSB se entiende que tiene tres ceros delante del 1 para formar un grupo de 4 bits.

Problema relacionado. Convertir el número hexadecimal 6BD3 a binario.

Debería estar claro que es mucho más fácil tratar con un número hexadecimal que con el número binario equivalente. Puesto que la conversión también es fácil, el sistema hexadecimal se usa ampliamente para representar los números binarios en programación, salidas de impresora y displays.

Conversión hexadecimal-decimal

Un método para encontrar el equivalente decimal de un número hexadecimal es, primero, convertir el número hexadecimal a binario, y después, el binario a decimal. El Ejemplo 2.26 ilustra este procedimiento.

EJEMPLO 2.26 Convertir a decimal los siguientes números hexadecimales:

(a) $1C_{16}$ **(b)** $A85_{16}$

Solución. En primer lugar, hay que convertir a binario el número hexadecimal, y después a decimal.

(a) 1 C → 00011100 $= 2^4 + 2^3 + 2^2 = 16 + 8 + 4 = \mathbf{28}_{10}$

(b) A 8 5 → 101010000101 $= 2^{11} + 2^9 + 2^7 + 2^2 + 2^0 = 2048 + 512 + 128 + 4 + 1 = \mathbf{2693}_{10}$

Problema relacionado. Convertir a decimal el número hexadecimal 6BD.

Otro método para convertir un número hexadecimal a su equivalente decimal es multiplicar el valor decimal de cada dígito hexadecimal por su peso, y luego realizar la suma de estos productos. Los pesos de un número hexadecimal crecen según las potencias de 16 (de derecha a izquierda). Para un número hexadecimal de 4 dígitos, los pesos son:

16^3	16^2	16^1	16^0
4096	256	16	1

El Ejemplo 2.27 muestra este método de conversión.

EJEMPLO 2.27 Convertir a decimal los siguientes números hexadecimales:

(a) $E5_{16}$ **(b)** $B2F8_{16}$

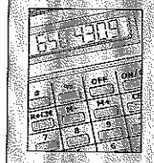
Solución. En la Tabla 2.3 puede verse que las letras de la A hasta a la F representan los números decimales de 10 hasta 15, respectivamente.

(a) $E5_{16} = (E \times 16) + (5 \times 1) = (14 \times 16) + (5 \times 1) = 224 + 5 = \mathbf{229}_{10}$

(b) $B2F8_{16} = (B \times 4096) + (2 \times 256) + (F \times 16) + (8 \times 1)$
$= (11 \times 4096) + (2 \times 256) + (15 \times 16) + (8 \times 1)$
$= 45.056 + 512 + 240 + 8 = \mathbf{45.816}_{10}$

Problema relacionado. Convertir a decimal el número hexadecimal $60A_{16}$.

Consejos para el uso de la calculadora



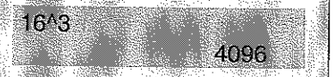
Las potencias de 16 se pueden obtener en la calculadora utilizando la tecla ^.

Ejemplo. Cálculo del valor de 16^3.

Paso 1. Presionar 1 6

Paso 2. Presionar ^

Paso 3. Presionar 3 y luego ENTER

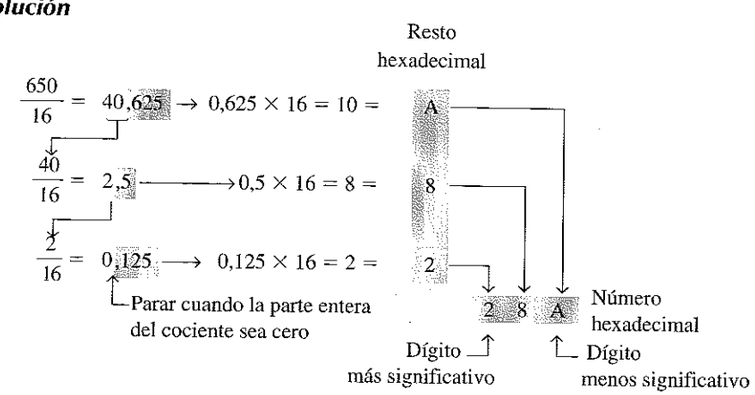


Conversión decimal-hexadecimal

La división sucesiva por 16 de un número decimal generará el número hexadecimal equivalente formado por los restos de las divisiones. El primer resto que se genera es el dígito menos significativo (LSD). Cada división sucesiva por 16 dará un resto que será un dígito del número hexadecimal equivalente. Este procedimiento es similar a la división sucesiva por 2 para la conversión decimal-binario, que se ha visto en la Sección 2.3. El Ejemplo 2.28 ilustra el procedimiento. Se observa que cuando un cociente tiene parte fraccionaria, ésta se multiplica por el divisor para obtener el resto.

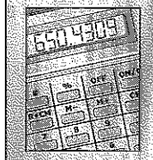
EJEMPLO 2.28 Convertir a hexadecimal el número decimal 650 por el método de la división sucesiva por 16.

Solución



Problema relacionado. Convertir a hexadecimal el número decimal 2591.

Consejos para el uso de la calculadora



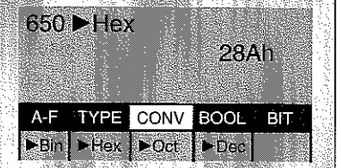
Un número decimal puede convertirse a hexadecimal usando la secuencia de teclas de conversión 2nd BASE F2.

Ejemplo. Conversión del decimal 650 a hexadecimal.

Paso 1. Presionar 6 5 0

Paso 2. Presionar F2

Paso 3. Presionar ENTER



Suma hexadecimal

La suma puede hacerse directamente con números hexadecimales, teniendo en cuenta que los dígitos hexadecimales de 0 a 9 son equivalentes a los dígitos decimales de 0 a 9, y que los dígitos hexadecimales de A hasta F son equivalentes a los números decimales 10 a 15. Cuando se suman dos números hexadecimales, se usan las siguientes reglas (los números decimales se indican con el subíndice 10).

1. En cualquier columna dada de una suma, pensar en los dos dígitos hexadecimales en términos de su valor decimal. Por ejemplo, $5_{16} = 5_{10}$ y $C_{16} = 12_{10}$.
2. Si la suma de los dos dígitos es 15_{10} o menor, reducir al dígito hexadecimal correspondiente.
3. Si la suma de los dígitos es mayor que 15_{10}, hay que reducir la suma que excede de 16_{10}, y pasar el acarreo de 1 a la siguiente columna.

EJEMPLO 2.29 Sumar los siguientes números hexadecimales:
(a) $23_{16} + 16_{16}$ **(b)** $58_{16} + 22_{16}$ **(c)** $2B_{16} + 84_{16}$ **(d)** $DF_{16} + AC_{16}$

Solución

(a)
$$\begin{array}{rl} 23_{16} & \text{columna derecha: } 3_{16} + 6_{16} = 3_{10} + 6_{10} = 9_{10} = 9_{16} \\ +\ 16_{16} & \text{columna izquierda: } 2_{16} + 1_{16} = 2_{10} + 1_{10} = 3_{10} = 3_{16} \\ \hline \mathbf{39}_{16} & \end{array}$$

(b)
$$\begin{array}{rl} 58_{16} & \text{columna derecha: } 8_{16} + 2_{16} = 8_{10} + 2_{10} = 10_{10} = A_{16} \\ +\ 22_{16} & \text{columna izquierda: } 5_{16} + 2_{16} = 5_{10} + 2_{10} = 7_{10} = 7_{16} \\ \hline \mathbf{7A}_{16} & \end{array}$$

(c)
$$\begin{array}{rl} 2B_{16} & \text{columna derecha: } B_{16} + 4_{16} = 11_{10} + 4_{10} = 15_{10} = F_{16} \\ +\ 84_{16} & \text{columna izquierda: } 2_{16} + 8_{16} = 2_{10} + 8_{10} = 10_{10} = A_{16} \\ \hline \mathbf{AF}_{16} & \end{array}$$

(d)
$$\begin{array}{rl} DF_{16} & \text{columna derecha: } F_{16} + C_{16} = 15_{10} + 12_{10} = 27_{10} \\ +\ AC_{16} & 27_{10} - 16_{10} = 11_{10} = B_{16} \text{ con acarreo 1} \\ \hline \mathbf{18B}_{16} & \text{columna izquierda: } D_{16} + A_{16} + 1_{16} = 13_{10} + 10_{10} + 1_{10} = 24_{10} \\ & 24_{10} - 16_{10} = 8_{10} = 8_{16} \text{ con acarreo 1} \end{array}$$

Problema relacionado. Sumar $4C_{16}$ y $3A_{16}$.

Sustracción hexadecimal utilizando el método del complemento a 2

Puesto que un número hexadecimal se puede usar para representar un número binario, también se puede emplear para representar el complemento a 2 del número binario. Por ejemplo, la representación hexadecimal de 11001001_2 es $C9_{16}$. El complemento a 2 de este número binario es 00110111, que en hexadecimal se escribe como 37_{16}.

Como ya se sabe, el complemento a 2 permite restar sumando números binarios. Luego también se puede usar este método para la sustracción en hexadecimal, como ilustra el Ejemplo 2.30.

EJEMPLO 2.30 Restar los siguientes números hexadecimales:
(a) $84_{16} - 2A_{16}$ **(b)** $C3_{16} - 0B_{16}$

Solución

(a) $2A_{16} = 00101010$
Complemento a 2 de $2A_{16} = 11010110 = D6_{16}$

84_{16}
$+ D6_{16}$ suma
$\not1 5A_{16}$ el acarreo no se tiene en cuenta, como en la suma en complemento a 2

La diferencia es $\mathbf{5A}_{16}$

(b) $0B_{16} = 00001011$
Complemento a 2 de $0B_{16} = 11110101 = F5_{16}$

$C3_{16}$
$+ F5_{16}$ suma
$\not1 B8_{16}$ el acarreo no se tiene en cuenta

La diferencia es $\mathbf{B8}_{16}$

Problema relacionado. Restar 173_{16} de BCD_{16}.

REVISIÓN DE LA SECCIÓN 2.8

1. Convertir a hexadecimal los números binarios:
   (a) 10110011 **(b)** 110011101000
2. Convertir a decimal los números hexadecimales:
   (a) 57_{16} **(b)** $3A5_{16}$ **(c)** $F80B_{16}$
3. Convertir a decimal $9B30_{16}$.
4. Convertir a hexadecimal el número decimal 573.
5. Sumar directamente los números hexadecimales:
   (a) $18_{16} + 34_{16}$ **(b)** $3F_{16} + 2A_{16}$
6. Restar los números hexadecimales:
   (a) $75_{16} - 21_{16}$ **(b)** $94_{16} - 5C_{16}$

2.9 ■ NÚMEROS OCTALES

Como el sistema hexadecimal, el sistema octal proporciona un método adecuado para expresar los códigos y números binarios. Sin embargo, se usa menos frecuentemente que el hexadecimal en las computadoras y microprocesadores para expresar magnitudes binarias con propósitos de entrada y salida.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Escribir los dígitos del sistema de numeración octal. ■ Convertir de octal a decimal. ■ Convertir de decimal a octal. ■ Convertir de octal a binario. ■ Convertir de binario a octal.

El sistema de numeración **octal** está formado por ocho dígitos, que son:

0, 1, 2, 3, 4, 5, 6, 7

Para contar por encima de 7, añadimos otra columna y continuamos así:

10, 11, 12, 13, 14, 15, 16, 17, 20, 21

y así sucesivamente. Contar en octal es parecido a contar en decimal, excepto que los dígitos 8 y 9 no se usan. Para distinguir los números octales de los números decimales y hexadecimales, utilizaremos el subíndice 8 para indicar un número octal. Por ejemplo, 15_8 es equivalente a 13_{10} en decimal y a D en hexadecimal. En ocasiones, puede verse una "o" o una "Q" detrás de un número octal.

Conversión octal-decimal

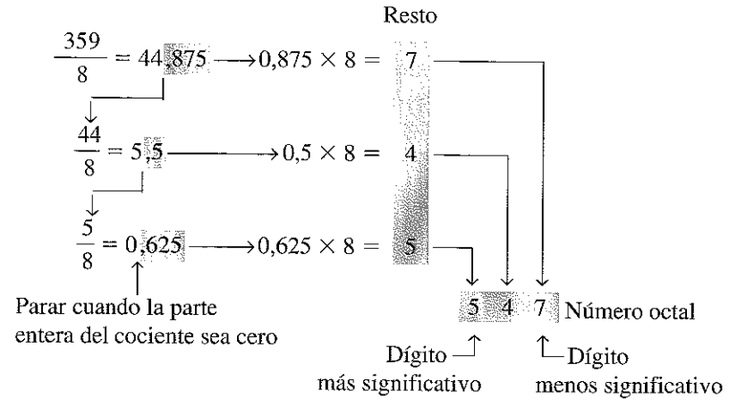
Puesto que el sistema de numeración octal es un sistema en base ocho, cada posición sucesiva de dígito es una potencia superior de ocho, empezando por el dígito situado más a la derecha con 8^0. La evaluación de un número octal en términos de su equivalente decimal se consigue multiplicando cada dígito por su peso y sumando los productos, como se muestra a continuación para 2374_8:

Peso: 8^3 8^2 8^1 8^0
Número octal: 2 3 7 4

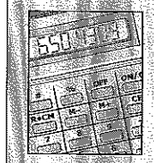
$$\begin{aligned} 2374_8 &= (2 \times 8^3) + (3 \times 8^2) + (7 \times 8^1) + (4 \times 8^0) \\ &= (2 \times 512) + (3 \times 64) + (7 \times 8) + (4 \times 1) \\ &= 1024 + 192 + 56 + 4 = 1276_{10} \end{aligned}$$

Conversión decimal-octal

Un método para convertir un número decimal en un número octal es el método de la división sucesiva por 8, que es parecido al método utilizado en la conversión a binario o a hexadecimal de los números decimales. Para mostrar cómo se hace, convertimos a octal el número decimal 359. Cada división sucesiva por 8 da un resto que será un dígito del número octal equivalente. El primer resto que se genera es el dígito menos significativo.



Consejos para el uso de la calculadora



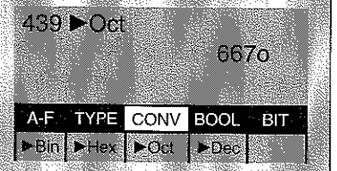
Un número decimal puede convertirse a octal usando la secuencia de teclas de conversión 2nd BASE F3

Ejemplo. Conversión del decimal 439 a octal.

Paso 1. Presionar 4 3 9

Paso 2. Presionar F3

Paso 3. Presionar ENTER



Conversión octal-binario

Puesto que cada dígito octal se puede representar mediante un número binario de 3 dígitos, es fácil convertir a binario un número octal. Cada dígito octal se representa mediante tres bits, como se muestra en la Tabla 2.4.

TABLA 2.4 *Conversión octal-binario*

Dígito octal	0	1	2	3	4	5	6	7
Binario	000	001	010	011	100	101	110	111

Para convertir un número octal en un número binario, simplemente se reemplaza cada dígito octal por el correspondiente grupo de tres bits. Este procedimiento se ilustra en el Ejemplo 2.31.

EJEMPLO 2.31 Convertir a binario los siguientes números octales:

(a) 13_8 **(b)** 25_8 **(c)** 140_8 **(d)** 7526_8

Solución

(a) 1 3 → **001 011**

(b) 2 5 → **010 101**

(c) 1 4 0 → **001 100 000**

(d) 7 5 2 6 → **111 101 010 110**

Problema relacionado. Convertir a decimal cada uno de los números binarios y verificar que los valores concuerdan con el valor decimal del número octal correspondiente.

Conversión binario-octal

La conversión de un número binario a un número octal es el inverso de la conversión de octal a binario. El procedimiento es el siguiente: se comienza por el grupo de tres bits más a la derecha y, moviéndose de derecha a izquierda, se convierte cada grupo de 3 bits en el dígito octal equivalente. Si para el grupo más a la izquierda no hay disponibles tres bits, se añaden uno o dos ceros para completar el grupo. Estos ceros no afectan al valor del número binario.

EJEMPLO 2.32 Convertir a octal los siguientes números binarios:

(a) 110101 **(b)** 101111001 **(c)** 100110011010 **(d)** 11010000100

Solución

(a) 110101 → 6 5 = $\mathbf{65_8}$

(b) 101111001 → 5 7 1 = $\mathbf{571_8}$

(c) 100110011010 → 4 6 3 2 = $\mathbf{4632_8}$

(d) 011010000100 → 3 2 0 4 = $\mathbf{3204_8}$

Problema relacionado. Convertir a octal el número binario 1010101000111110010.

REVISIÓN DE LA SECCIÓN 2.9

1. Convertir a decimal los números octales:
   (a) 73_8 **(b)** 125_8
2. Convertir a octal los números decimales:
   (a) 98_{10} **(b)** 163_{10}
3. Convertir a binario los números octales:
   (a) 46_8 **(b)** 723_8 **(c)** 5624_8
4. Convertir a octal los números binarios:
   (a) 110101111 **(b)** 1001100010 **(c)** 10111111001

2.10 ■ CÓDIGO DECIMAL BINARIO (BCD)

El código decimal binario (BCD Binary Coded Decimal) es una forma de expresar cada uno de los dígitos decimales con un código binario. Puesto que en el sistema BCD sólo existen diez grupos de código, es muy fácil convertir entre decimal y BCD. Como nosotros leemos y escribimos en decimal, el código BCD proporciona una excelente interfaz para los sistemas binarios. Ejemplos de estas interfaces son las entradas por teclado y las salidas digitales.

Al finalizar esta sección, el lector deberá ser capaz de:

- Convertir cada dígito decimal a BCD.
- Expresar en BCD números decimales.
- Convertir de BCD a decimal.
- Sumar números en BCD.

El código 8421

El código 8421 es un tipo de **código decimal binario (BCD)**. Código decimal binario significa que cada dígito decimal, de 0 hasta 9, se representa mediante un **código** binario de cuatro bits. La designación 8421 indica los pesos binarios de los cuatro bits (2^3, 2^2, 2^1, 2^0). La facilidad de conversión entre los números en código 8421 y los familiares números decimales es la principal ventaja de este código. Todo lo que hay que recordar sobre las diez combinaciones binarias que representan los diez dígitos decimales se muestra en la Tabla 2.5. El código 8421 es el código BCD más importante, y cuando hacemos referencia a BCD, siempre es al código 8421, a no ser que se indique otra cosa.

TABLA 2.5
Conversion decimal/BCD

Dígito decimal	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Códigos no válidos. Deberíamos saber que, con cuatro dígitos, se pueden representar dieciséis números (desde 0000 hasta 1111), pero en el código 8421, sólo se usan diez de ellos. Las seis combinaciones que no se emplean (1010, 1011, 1100, 1101, 1110 y 1111) no son válidas en el código BCD 8421.

Para expresar cualquier número decimal en BCD, simplemente reemplazar cada dígito decimal por el apropiado código de 4 bits, tal como muestra el Ejemplo 2.33.

EJEMPLO 2.33 Convertir a BCD los siguientes números decimales:
(a) 35 **(b)** 98 **(c)** 170 **(d)** 2469

Solución

(a) 3 5 → **00110101**
(b) 9 8 → **10011000**
(c) 1 7 0 → **000101110000**
(d) 2 4 6 9 → **0010010001101001**

Problema relacionado. Convertir a BCD el número decimal 9673.

Es igualmente sencillo determinar el número decimal a partir del código BCD. Se comienza por el bit más a la derecha y se divide el código en grupos de cuatro bits. Después se escribe el dígito decimal representado por cada grupo de 4 bits. El Ejemplo 2.34 ilustra esto.

EJEMPLO 2.34 Convertir a decimal los siguientes códigos BCD:
(a) 10000110 **(b)** 001101010001 **(c)** 1001010001110000

Solución

(a) 10000110 → **8 6**
(b) 001101010001 → **3 5 1**
(c) 1001010001110000 → **9 4 7 0**

Problema relacionado. Convertir a decimal el código BCD 1000010001001110110.

Suma en BCD

BCD es un código numérico y puede utilizarse en operaciones aritméticas. La suma es la más importante de estas operaciones, ya que las otras tres operaciones (sustracción, multiplicación y división) se pueden llevar a cabo utilizando la suma. A continuación, vamos a ver cómo se suman dos números BCD:

Paso 1. Sumar los dos números BCD utilizando las reglas de la suma binaria vistas en la Sección 2.4.
Paso 2. Si una suma de 4 bits es igual o menor que 9, es un número BCD válido.
Paso 3. Si una suma de 4 bits es mayor que 9, o si genera un acarreo en el grupo de 4 bits, el resultado no es válido. En este caso, se suma 6 (0110) al grupo de 4 bits para saltar así los seis estados no válidos y pasar al código 8421. Si se genera un acarreo al sumar 6, éste se suma al grupo de 4 bits siguiente.

El Ejemplo 2.35 ilustra la suma en BCD para los casos en que la suma en cada columna de 4 bits es igual o menor que 9 y, por tanto, las sumas de 4 bits son números BCD válidos. El Ejemplo 2.36 ilustra el procedimiento en el caso de que se produzcan sumas no válidas (mayores que 9 o que generen acarreo).

EJEMPLO 2.35 Sumar los siguientes números BCD:
(a) 0011 + 0100 **(b)** 00100011 + 00010101
(c) 10000110 + 00010011 **(d)** 010001010000 + 010000010111

Solución. Se muestra la suma en decimal con propósitos de comparación.

(a)
```
  0011      3
+ 0100    + 4
------    ---
  0111      7
```

(b)
```
  0010   0011     23
+ 0001   0101   + 15
------   ----   ----
  0011   1000     38
```

(c)	1000	0110	86	**(d)**	0100	0101	0000	450
	+ 0001	0011	+ 13		+ 0100	0001	0111	+ 417
	1001	**1001**	99		**1000**	**0110**	**0111**	867

Observe que en ningún caso la suma de las cuatro columnas de 4 bits excede 9, por lo que los resultados son números BCD válidos.

Problema relacionado. Sumar los números BCD: 1001000001000011 y 0000100100100101.

EJEMPLO 2.36 Sumar los siguientes números BCD:

(a) 1001 + 0100 **(b)** 1001 + 1001
(c) 00010110 + 00010101 **(d)** 01100111 + 01010011

Solución. La suma en números decimales se indica con propósitos de comparación.

(a)

	1001		9
	+ 0100		+ 4
	1101	Número BCD no válido (>9)	13
	+ 0110	Se suma 6	
0001	**0011**	Número BCD válido	
↓	↓		
1	3		

(b)

	1001		9
	+ 1001		+ 9
1	0010	No válido debido al acarreo	18
	+ 0110	Se suma 6	
0001	**1000**	Número BCD válido	
↓	↓		
1	8		

(c)

0001	0110		16
+ 0001	0101		+ 15
0010	1011	El grupo de la derecha no es válido (>9), el grupo de la izquierda si es válido.	31
	+ 0110	Se suma 6 al código no válido. Se suma el acarreo, 0001, al siguiente grupo.	
0011	**0001**	Número BCD válido	
↓	↓		
3	1		

(d)

0110	0111		67
+ 0101	0011		+ 53
1011	1010	Ambos grupos no son válidos (>9)	120
+ 0110	+ 0110	Se suma 6 a ambos grupos	
0001	**0010**	**0000**	Número BCD válido
↓	↓	↓	
1	2	0	

Problema relacionado. Sumar los números BCD: 01001000 + 00110100.

REVISIÓN DE LA SECCIÓN 2.10

1. ¿Cuál es el peso binario de cada uno de los 1s en los siguientes números BCD?
   (a) 0010 **(b)** 1000 **(c)** 0001 **(d)** 0100
2. Convertir a BCD los siguientes números decimales:
   (a) 6 **(b)** 15 **(c)** 273 **(d)** 849
3. ¿Qué números decimales representan los códigos BCD siguientes?
   (a) 10001001 **(b)** 001001111000 **(c)** 000101010111
4. En la suma BCD, ¿cuándo no es válida una suma de 4 bits?

2.11 ■ CÓDIGOS DIGITALES Y PARIDAD

Existen muchos códigos especializados que se usan en los sistemas digitales. Acabamos de aprender el código BCD; ahora vamos a ver algunos otros. Algunos códigos son estrictamente numéricos, como BCD, y otros son alfanuméricos, es decir, se utilizan para representar números, letras, símbolos e instrucciones. En esta sección se presentan el código Gray y el código ASCII. También se cubre la detección de errores en los códigos mediante el bit de paridad.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar la ventaja del código Gray. ■ Convertir entre código Gray y código binario. ■ Utilizar el código ASCII. ■ Identificar los errores en los códigos basándose en el método de paridad.

El código Gray

El **código Gray** es un código sin pesos y no aritmético, es decir, no existen pesos específicos asignados a las posiciones de los bits. La característica más importante del código Gray es que *sólo varía un bit de un código al siguiente*. Esta propiedad es importante en muchas aplicaciones, tales como los codificadores de eje de posición, en los que la susceptibilidad de error aumenta con el número de cambios de bit entre números adyacentes dentro de una secuencia.

La Tabla 2.6 presenta el código Gray de cuatro bits para los números decimales de 0 a 15. Como referencia se muestran también en la tabla los números binarios. Como en los números binarios, *el código Gray puede tener cualquier número de bits*. Obsérvese que, en este código, sólo cambia un bit entre los sucesivos números. Por ejemplo, para pasar del decimal 3 al 4, el código Gray lo hace de 0010 a 0110, mientras que el código binario lo hace de 0011 a 0100, cambiando tres bits. En el código Gray, el único bit que cambia es el tercer bit de la derecha, los restantes permanecen igual.

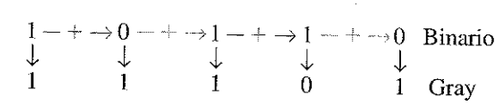
Conversión de código binario a código Gray. Algunas veces, la conversión de código binario a código Gray resulta útil. Las siguientes reglas explican cómo convertir un número binario en un número en código Gray:

1. El bit más significativo (el que está más a la izquierda, MSB) en el código Gray es el mismo que el correspondiente MSB del número binario.
2. Yendo de izquierda a derecha, sumar cada par adyacente de los bits en código binario para obtener el siguiente bit en código Gray. Los acarreos deben descartarse.

TABLA 2.6
Código Gray de cuatro bits.

Decimal	Binario	Código Gray	Decimal	Binario	Código Gray
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000

Por ejemplo, la conversión del número binario 10110 a código Gray se hace del siguiente modo:

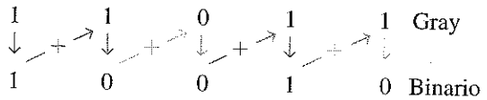


El código Gray es 11101.

Conversión de Gray a binario. Para convertir de código Gray a binario, se utiliza un método similar, pero con algunas diferencias. Se aplican las siguientes reglas:

1. El bit más significativo (bit más a la izquierda) en el código binario es el mismo que el correspondiente bit en código Gray.
2. A cada bit del código binario generado se le suma el bit en código Gray de la siguiente posición adyacente. Los acarreos se descartan.

Por ejemplo, la conversión del número en código Gray 11011 a binario es como sigue:

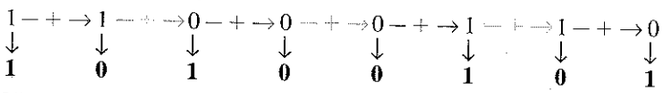


El número binario es 10010.

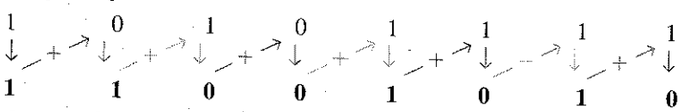
EJEMPLO 2.37 **(a)** Convertir a código Gray el número binario 11000110.
(b) Convertir a binario el código Gray 10101111.

Solución

(a) Binario a código Gray:



(b) Código Gray a binario:



Problema relacionado. **(a)** Convertir a código Gray el número binario 101101. **(b)** Convertir a binario el código Gray 100111.

Ejemplo de aplicación

En la Figura 2.4 se muestra un diagrama simplificado de un mecanismo codificador de eje de 3 bits. Básicamente, está formado por tres anillos concéntricos conductores que están divididos en ocho sectores. Cuantos más sectores haya, con mayor precisión se puede representar la posición, pero aquí sólo se utilizan ocho con propósitos de ilustración. Cada sector de cada anillo se fija a un nivel alto o a un nivel bajo de tensión para representar un 1 o un 0. Los sectores de color gris indican los 1s, y los sectores blancos los 0s. Puesto que los anillos giran con el eje, hacen contacto con una disposición de escobillas que están en una posición fija, y a las que están conectadas las líneas de salida. Como el eje gira en el sentido contrario a las agujas del reloj a través de los 360°, los ocho sectores se mueven y pasan por las tres escobillas produciendo una salida binaria de tres bits que indica la posición del eje.

En la Figura 2.4(a), los sectores están colocados según el modelo binario normal, por lo que las escobillas van de 000 a 001, de 010 a 011, y así sucesivamente. Cuando las escobillas se encuentran sobre los sectores de color gris, la salida es 1, y cuando están sobre los sectores blancos, la salida es 0. Si una de las escobillas, durante la transición de un sector al siguiente, se adelanta ligeramente, puede producirse una salida errónea. Consideremos lo que ocurre cuando las escobillas están en el sector 111, y pasan al sector 000. Si la escobilla MSB se adelanta ligeramente, la posición se indicaría de forma incorrecta por una transición a 011 en lugar de 111 a 000. En este tipo de aplicaciones, es virtualmente imposible mantener el alineamiento mecánico preciso de todas las escobillas, por lo que siempre se producirán algunos errores en muchas de las transiciones entre sectores.

El código Gray se utiliza para eliminar los errores que son inherentes al código binario. Como se muestra en la Figura 2.4(b), el código Gray asegura que sólo un bit variará entre sectores adyacentes. Esto significa que aunque las escobillas no puedan tener un alineamiento preciso, nunca se producirá un error de transición. Por ejemplo, consideremos de nuevo qué ocurre cuando las escobillas están en el sector 111 y se mueven al sector siguiente, 101. Las dos únicas posibles salidas durante la transición son 111 y 101, independientemente de cómo estén alineadas las escobillas. Una situación similar se produce en las transiciones entre los restantes sectores.

Códigos alfanuméricos

Para la comunicación, no sólo se necesitan números, sino también letras y otros símbolos. En sentido estricto, los códigos **alfanuméricos** son códigos que representan números y caracteres alfabéticos (letras). Sin embargo, la mayoría de estos códigos también representan otros caracteres tales como símbolos y distintas instrucciones necesarias para la transferencia de información.

Como mínimo, un código alfanumérico debe poder representar los diez dígitos decimales y las 26 letras del alfabeto, es decir, un total de 36 elementos. Esta cantidad requiere seis bits para cada combinación de código, puesto que cinco son insuficientes ($2^5 = 32$). Con seis bits se tiene un total de 64 combinaciones, por lo que 28 de ellas no se utilizan. Obviamente, en muchas aplicaciones, para completar la comunicación, son necesarios otros símbolos además de los números y las letras. Se necesitan espacios, punto, dos puntos, punto y coma, signo de interrogación, etcétera. También se necesitan instrucciones para comunicar al sistema receptor qué hacer con la información. De este modo, con códigos con una longitud de seis bits, se pueden manejar

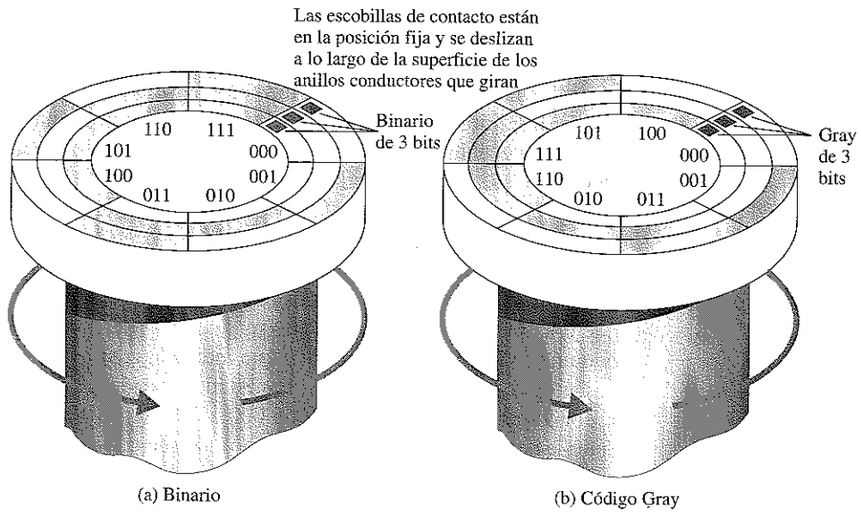


FIGURA 2.4
Ilustración simplificada de cómo el código Gray resuelve el problema de error en los codificadores de posición de eje.

NOTAS INFORMÁTICAS
Un teclado de computadora tiene un microprocesador dedicado, que explora constantemente los circuitos de teclado para detectar cuándo se ha presionado y liberado una tecla. Un paquete de software genera un código de exploración unívoco que representa a dicha tecla en particular. El código de exploración se convierte a continuación en un código alfanumérico (ASCII) que utilizará la computadora.

números decimales, el alfabeto y otros 28 símbolos. Esto debería dar una idea de los requerimientos para un código alfanumérico básico. El ASCII es el código alfanumérico más común y se va a tratar a continuación.

ASCII

El **American Standard Code for Information Interchange** (ASCII, código estándar americano para el intercambio de información) es un código alfanumérico universalmente aceptado, que se usa en la mayoría de las computadoras y otros equipos electrónicos. La mayor parte de los teclados de computadora se estandarizan de acuerdo con el código ASCII, y cuando se pulsa una letra, un número o un comando de control, es el código ASCII el que se introduce en la computadora.

El código ASCII dispone de 128 caracteres que se representan mediante un código binario de 7 bits. Realmente, el código ASCII puede considerarse como un código de 8 bits en el que el MSB siempre es 0. En hexadecimal, este código de 8 bits va de 00 hasta 7F. Los primeros 32 caracteres ASCII son comandos no gráficos, que nunca se imprimen o presentan en pantalla, y sólo se utilizan para propósitos de control. Ejemplos de caracteres de control son el carácter nulo, avance de línea, inicio de texto y escape. Los demás caracteres son símbolos gráficos que pueden imprimirse o mostrarse en pantalla, e incluyen las letras del alfabeto (mayúsculas y minúsculas), los diez dígitos decimales, los signos de puntuación y otros símbolos comúnmente utilizados.

La Tabla 2.7 presenta un listado del código ASCII, con su representación decimal, hexadecimal y binaria para cada carácter y símbolo. En la primera columna de la tabla se enumeran los nombres de los 32 caracteres de control (en hexadecimal, de 00 hasta 1F), y en las restantes columnas se muestran los símbolos gráficos (en hexadecimal, de 20 hasta 7F).

EJEMPLO 2.38 Determinar los códigos binarios ASCII que se han introducido a través del teclado de la computadora cuando se ha escrito la instrucción BASIC siguiente. Expresar también cada código en hexadecimal.

```
20 PRINT "A=";X
```

Solución. En la Tabla 2.7 puede encontrar el código ASCII correspondiente a cada carácter.

Carácter	*Binario*	*Hexadecimal*
2	0110010	32_{16}
0	0110000	30_{16}
Space	0100000	20_{16}
P	1010000	50_{16}
R	1010010	52_{16}
I	1001001	49_{16}
N	1001110	$4E_{16}$
T	1010100	54_{16}
Space	0100000	20_{16}
"	0100010	22_{16}
A	1000001	41_{16}
=	0111101	$3D_{16}$
"	0100010	22_{16}
;	0111011	$3B_{16}$
X	1011000	58_{16}

Problema relacionado. Determinar la secuencia de los códigos ASCII requerida para la siguiente instrucción de programa y expresarla en hexadecimal:

```
80 INPUT Y
```

Los caracteres de control ASCII. Los primeros treinta y dos códigos de la tabla ASCII (Tabla 2.7) representan los caracteres de control. Estos se utilizan para permitir a dispositivos, tales como una computadora o una impresora, que se comuniquen entre sí cuando transfieren información y datos. La Tabla 2.8 enumera los caracteres de control y las funciones de las teclas de control que permiten introducir directamente el código ASCII a través del teclado, presionando la tecla control (CTRL) y el símbolo correspondiente. También se facilita una breve descripción de cada carácter de control.

Caracteres del código ASCII extendido

Además de los 128 caracteres ASCII estándar, existen 128 caracteres adicionales que fueron adoptados por IBM para utilizar en sus computadoras personales (PC). Debido a la popularidad del PC, estos caracteres especiales del código ASCII extendido se usan también en otras aplicaciones distintas de los PC, por lo que se ha convertido en un estándar no oficial.

Los caracteres del código ASCII extendido se representan mediante una serie de códigos de 8 bits que van, en hexadecimal, del 80 hasta FF.

TABLA 2.7
American Standard Code for Information Interchange (ASCII)

Caracteres de control				Símbolos gráficos				Símbolos gráficos			
Nombre	Dec	Binario	Hex	Símbolo	Dec	Binario	Hex	Símbolo	Dec	Binario	Hex
NUL	0	0000000	00	space	32	0100000	20	@	64	1000000	40
SOH	1	0000001	01	!	33	0100001	21	A	65	1000001	41
STX	2	0000010	02	"	34	0100010	22	B	66	1000010	42
ETX	3	0000011	03	#	35	0100011	23	C	67	1000011	43
EOT	4	0000100	04	$	36	0100100	24	D	68	1000100	44
ENQ	5	0000101	05	%	37	0100101	25	E	69	1000101	45
ACK	6	0000110	06	&	38	0100110	26	F	70	1000110	46
BEL	7	0000111	07	'	39	0100111	27	G	71	1000111	47
BS	8	0001000	08	(	40	0101000	28	H	72	1001000	48
HT	9	0001001	09	)	41	0101001	29	I	73	1001001	49
LF	10	0001010	0A	*	42	0101010	2A	J	74	1001010	4A
VT	11	0001011	0B	+	43	0101011	2B	K	75	1001011	4B
FF	12	0001100	0C	,	44	0101100	2C	L	76	1001100	4C
CR	13	0001101	0D	-	45	0101101	2D	M	77	1001101	4D
SO	14	0001110	0E	.	46	0101110	2E	N	78	1001110	4E
SI	15	0001111	0F	/	47	0101111	2F	O	79	1001111	4F
DLE	16	0010000	10	0	48	0110000	30	P	80	1010000	50
DC1	17	0010001	11	1	49	0110001	31	Q	81	1010001	51
DC2	18	0010010	12	2	50	0110010	32	R	82	1010010	52
DC3	19	0010011	13	3	51	0110011	33	S	83	1010011	53
DC4	20	0010100	14	4	52	0110100	34	T	84	1010100	54
NAK	21	0010101	15	5	53	0110101	35	U	85	1010101	55
SYN	22	0010110	16	6	54	0110110	36	V	86	1010110	56
ETB	23	0010111	17	7	55	0110111	37	W	87	1010111	57
CAN	24	0011000	18	8	56	0111000	38	X	88	1011000	58
EM	25	0011001	19	9	57	0111001	39	Y	89	1011001	59
SUB	26	0011010	1A	:	58	0111010	3A	Z	90	1011010	5A
ESC	27	0011011	1B	;	59	0111011	3B	[	91	1011011	5B
FS	28	0011100	1C	<	60	0111100	3C	\	92	1011100	5C
GS	29	0011101	1D	=	61	0111101	3D	]	93	1011101	5D
RS	30	0011110	1E	>	62	0111110	3E	^	94	1011110	5E
US	31	0011111	1F	?	63	0111111	3F	_	95	1011111	5F
									96	1100000	60
				a	97	1100001	61				
				b	98	1100010	62				
				c	99	1100011	63				
				d	100	1100100	64				
				e	101	1100101	65				
				f	102	1100110	66				
				g	103	1100111	67				
				h	104	1101000	68				
				i	105	1101001	69				
				j	106	1101010	6A				
				k	107	1101011	6B				
				l	108	1101100	6C				
				m	109	1101101	6D				
				n	110	1101110	6E				
				o	111	1101111	6F				
				p	112	1110000	70				
				q	113	1110001	71				
				r	114	1110010	72				
				s	115	1110011	73				
				t	116	1110100	74				
				u	117	1110101	75				
				v	118	1110110	76				
				w	119	1110111	77				
				x	120	1111000	78				
				y	121	1111001	79				
				z	122	1111010	7A				
				{	123	1111011	7B				
				|	124	1111100	7C				
				}	125	1111101	7D				
				~	126	1111110	7E				
				Del	127	1111111	7F				

TABLA 2.8
Caracteres de control ASCII

Nombre	Decimal	Hex	Tecla	Descripción
NUL	0	00	CTRL @	Carácter nulo
SOH	1	01	CTRL A	Inicio de cabecera
STX	2	02	CTRL B	Inicio de texto
ETX	3	03	CTRL C	Fin de texto
EOT	4	04	CTRL D	Fin de transmisión
ENQ	5	05	CTRL E	Petición
ACK	6	06	CTRL F	Reconocimiento
BEL	7	07	CTRL G	Timbre
BS	8	08	CTRL H	Barra espaciadora
HT	9	09	CTRL I	Tabulador horizontal
LF	10	0A	CTRL J	Avance de línea
VT	11	0B	CTRL K	Tabulador vertical
FF	12	0C	CTRL L	Salto de página
CR	13	0D	CTRL M	Retorno de carro
SO	14	0E	CTRL N	Desplazamiento de salida
SI	15	0F	CTRL O	Desplazamiento de entrada
DLE	16	10	CTRL P	Escape de enlace de datos
DC1	17	11	CTRL Q	Dispositivo de control 1
DC2	18	12	CTRL R	Dispositivo de control 2
DC3	19	13	CTRL S	Dispositivo de control 3
DC4	20	14	CTRL T	Dispositivo de control 4
NAK	21	15	CTRL U	Confirmación negativa
SYN	22	16	CTRL V	Sincronismo
ETB	23	17	CTRL W	Fin del bloque de transmisión
CAN	24	18	CTRL X	Cancelación
EM	25	19	CTRL Y	Fin del dispositivo
SUB	26	1A	CTRL Z	Sustitución
ESC	27	1B	CTRL [	Escape
FS	28	1C	CTRL /	Separador de fichero
GS	29	1D	CTRL]	Separador de grupo
RS	30	1E	CTRL ^	Separador de registro
US	31	1F	CTRL _	Separador de unidad

El código ASCII extendido está formado por caracteres que pertenecen a las siguientes categorías generales:

1. Caracteres alfabéticos no ingleses
2. Símbolos de moneda no ingleses
3. Letras griegas
4. Símbolos matemáticos
5. Caracteres para gráficos
6. Caracteres para gráficos de barras
7. Caracteres sombreados

En la Tabla 2.9 se enumera el conjunto de caracteres del código ASCII extendido, junto con sus representaciones decimal y hexadecimal.

Método de paridad para la detección de errores

Muchos sistemas emplean un bit de paridad como medio para la **detección de errores** de bit. Cualquier grupo de bits contiene un número par o impar de 1s. Un bit de **paridad** se añade al grupo de bits para hacer que el número total de 1s en el grupo sea siempre par o siempre impar. Un bit de paridad par hace que el número total de 1s sea par, y un bit de paridad impar hace que el número total de 1s del grupo sea impar.

Un determinado sistema puede funcionar con paridad par o impar, pero no con ambas. Por ejemplo, si un sistema trabaja con paridad par, una comprobación que se realice en cada grupo de bits recibidos tiene que asegurar que el número total de 1s en ese grupo es par. Si hay un número impar de 1s, quiere decir que se ha producido un error.

Vamos a ver cómo se asocian los bits de paridad a un código. La Tabla 2.10 enumera los bits de paridad de cada número BCD, tanto en el caso de paridad par como de paridad impar. El bit de paridad para cada número BCD se indica en la columna *P*.

TABLA 2.9
Caracteres de código ASCII extendido.

Símbolo	Dec	Hex	Símbolo	Dec	Hex	Símbolo	Dec	Hex	Símbolo	Dec	Hex
Ç	128	80	á	160	A0	└	192	C0	α	224	E0
ü	129	81	í	161	A1	┴	193	C1	β	225	E1
é	130	82	ó	162	A2	┬	194	C2	Γ	226	E2
â	131	83	ú	163	A3	├	195	C3	π	227	E3
ä	132	84	ñ	164	A4	─	196	C4	Σ	228	E4
à	133	85	Ñ	165	A5	┼	197	C5	σ	229	E5
å	134	86	a	166	A6	╞	198	C6	μ	230	E6
ç	135	87	o	167	A7	╟	199	C7	τ	231	E7
ê	136	88	¿	168	A8	╚	200	C8	Φ	232	E8
ë	137	89	⌐	169	A9	╔	201	C9	Θ	233	E9
è	138	8A	¬	170	AA	╩	202	CA	Ω	234	EA
ï	139	8B	½	171	AB	╦	203	CB	δ	235	EB
î	140	8C	¼	172	AC	╠	204	CC	∞	236	EC
ì	141	8D	¡	173	AD	═	205	CD	φ	237	ED
Ä	142	8E	«	174	AE	╬	206	CE	ε	238	EE
Å	143	8F	»	175	AF	╧	207	CF	∩	239	EF
É	144	90	▒	176	B0	╨	208	D0	≡	240	F0
æ	145	91	▒	177	B1	╤	209	D1	±	241	F1
Æ	146	92	▒	178	B2	╥	210	D2	≥	242	F2
ô	147	93	│	179	B3	╙	211	D3	≤	243	F3
ö	148	94	┤	180	B4	╘	212	D4	⌠	244	F4
ò	149	95	╡	181	B5	╒	213	D5	⌡	245	F5
û	150	96	╢	182	B6	╓	214	D6	÷	246	F6
ù	151	97	╖	183	B7	╫	215	D7	≈	247	F7
ÿ	152	98	╕	184	B8	╪	216	D8	°	248	F8
Ö	153	99	╣	185	B9	┘	217	D9	•	249	F9
Ü	154	9A	║	186	BA	┌	218	DA	·	250	FA
¢	155	9B	╗	187	BB	█	219	DB	√	251	FB
£	156	9C	╝	188	BC	▄	220	DC	η	252	FC
¥	157	9D	╜	189	BD	▌	221	DD	²	253	FD
P_t	158	9E	╛	190	BE	▐	222	DE	■	254	FE
f	159	9F	┐	191	BF	▀	223	DF		255	FF

TABLA 2.10
El código BCD con bits de paridad

Paridad par		Paridad impar	
P	BCD	*P*	BCD
0	0000	1	0000
1	0001	0	0001
1	0010	0	0010
0	0011	1	0011
1	0100	0	0100
0	0101	1	0101
0	0110	1	0110
1	0111	0	0111
1	1000	0	1000
0	1001	1	1001

El bit de paridad se puede añadir al principio o al final del código, dependiendo del diseño del sistema. Obsérvese que el número total de 1s, incluyendo el bit de paridad, siempre es par para paridad par, y siempre es impar para paridad impar.

Detección de un error. Un bit de paridad facilita la detección de un único error de bit (o de cualquier número impar de errores, lo cual es muy improbable), pero no puede detectar dos errores dentro de un grupo. Por ejemplo, supongamos que deseamos transmitir el código BCD 0101 (el método de paridad puede usarse con cualquier número de bits, ahora usamos cuatro con propósitos de ilustración). El código total transmitido incluyendo el bit de paridad par es

```
Bit de paridad par
↓
00101
  ↑
Código BCD
```

Ahora, supongamos que se ha producido un error en el tercer bit de la izquierda (el 1 se transmite como 0) como sigue:

```
Bit de paridad par
↓
00001
  ↑
Bit erróneo
```

Cuando se recibe este código, la circuitería de comprobación de paridad determina que sólo hay un 1 (impar), cuando debería haber un número par de 1s. Puesto que, en el código recibido no aparece un número par de 1s, esto indica que se ha producido un error.

Un bit de paridad impar también facilita de forma similar la detección de un único error en un grupo de bits dado.

EJEMPLO 2.39 Asignar el bit de paridad par apropiado a los siguientes grupos de código:

(a) 1010 **(b)** 111000 **(c)** 101101
(d) 100011100101 **(e)** 101101011111

Solución. El bit de paridad par debe ser 1 o 0 de modo que el número total de 1s sea par. El bit de paridad será el bit más a la izquierda (tramado).

(a) 0 1010 **(b)** 1 111000 **(c)** 0 101101

(d) 0 100011100101 **(e)** 1 101101011111

Problema relacionado. Añadir un bit de paridad par al código ASCII de 7 bits correspondiente a la letra K.

EJEMPLO 2.40 Un sistema de paridad impar recibe los siguientes grupos de códigos: 10110, 11010, 110011, 110101110100 y 1100010101010. Determinar qué grupos, si hay alguno, tienen error.

Solución. Puesto que se requiere paridad impar, cualquier grupo con un número par de 1s es incorrecto. Los siguientes grupos contienen error: **110011** y **1100010101010**.

Problema relacionado. En un sistema de paridad impar se recibe el siguiente carácter ASCII: 00110111. ¿Es correcto?

REVISIÓN DE LA SECCIÓN 2.11

1. Convertir a código Gray los siguientes números binarios:
   (a) 1100 **(b)** 1010 **(c)** 11010
2. Convertir a binario los siguientes números en código Gray:
   (a) 1000 **(b)** 1010 **(c)** 11101
3. ¿Cuál es la representación ASCII de cada uno de los siguientes caracteres? Expresarlos como grupo de bits y en notación hexadecimal.
   (a) K **(b)** r **(c)** $ **(d)** +
4. Añadir un bit de paridad par a cada código ASCII de la cuestión 3.

2.12 ■ APLICACIÓN A LOS SISTEMAS DIGITALES

El sistema presentado en el Capítulo 1 facilita un método sencillo para ilustrar cómo los códigos y números binarios se pueden utilizar en un sistema digital. Recuérdese que el sistema controla el número de pastillas de cada bote que se encuentra en una línea de montaje de una planta farmacéutica. En esta sección, veremos un ejemplo de los códigos y números binarios que existen en varios puntos del sistema durante una fase del proceso. También determinará los códigos y números a través del sistema bajo ciertas condiciones específicas.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar la finalidad de cada bloque funcional del sistema, en términos de códigos y números. ■ Analizar el sistema bajo condiciones específicas, en términos de números y códigos en determinados puntos.

Funcionamiento del sistema

Los diagramas de bloques de las Figuras 2.5, 2.6 y 2.7 muestran el estado del sistema en determinados puntos del proceso. Los números binarios (tramados) y códigos BCD (en negrita) indican las entradas y salidas de cada bloque, y los contenidos de los dispositivos de almacenamiento. El valor decimal de cada código binario o BCD se especifica entre paréntesis. Las flechas gruesas representan líneas paralelas que conectan los distintos bloques del sistema, y proporcionan caminos para la transferencia en paralelo de datos. Por ejemplo, ocho bits de un número binario dado se presentan simultáneamente sobre las ocho líneas paralelas. Las líneas paralelas tales como éstas, comúnmente se denominan *buses* en la terminología de sistemas digitales.

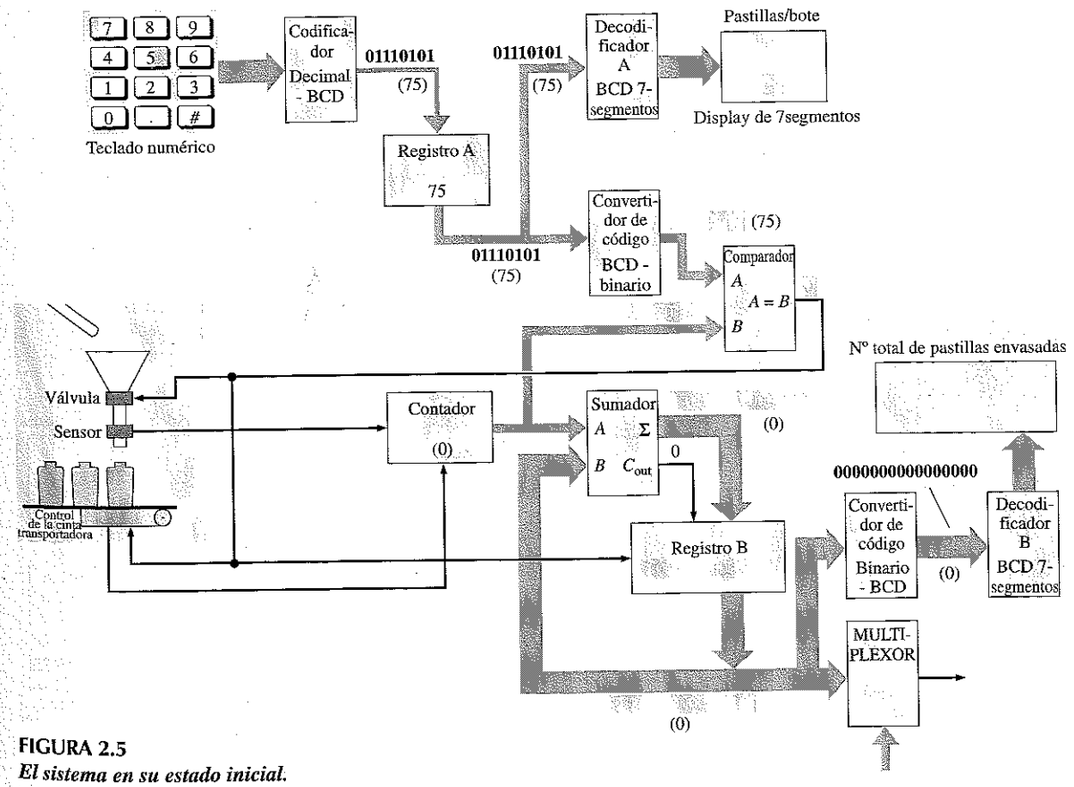


FIGURA 2.5
El sistema en su estado inicial.

Es importante tener en cuenta que el diagrama de bloques mostrado está en formato simplificado, y se ha prescindido de muchos detalles, que en este momento no son esenciales. También se ha omitido la unidad remota que se mostró en el Capítulo 1, para hacer las cosas tan sencillas como sea posible.

Inicialización del sistema. Antes de nada, el sistema se debe inicializar para arrancar en los estados binarios correctos. En primer lugar, el contador y los registros

se deben poner a cero, para que la cuenta empiece a partir de cero. A continuación, el número de pastillas (hasta 99) que se introducirá en cada bote, se introduce a través del teclado numérico, el cual se convierte de decimal a BCD mediante el codificador, y se almacena en el registro A.

En la Figura 2.5, el número decimal 75 se introduce a través del teclado numérico, lo que indica que en cada bote se introducirán 75 pastillas. Cada pulsación de teclado se codifica en BCD y el dato se envía, los 4 bits a la vez, al registro A, para ser almacenado. El código BCD de 8 bits se coloca en las líneas de salida del registro y se aplica al decodificador, el cual lo convierte de BCD a código de 7 segmentos para iluminar el display. Simultáneamente, el mismo código BCD se aplica al convertidor de código, el cual lo convierte a binario. El número binario resultante, 75, se presenta en la entrada *A* del comparador. Ahora el sistema está inicializado y listo para empezar a contar pastillas.

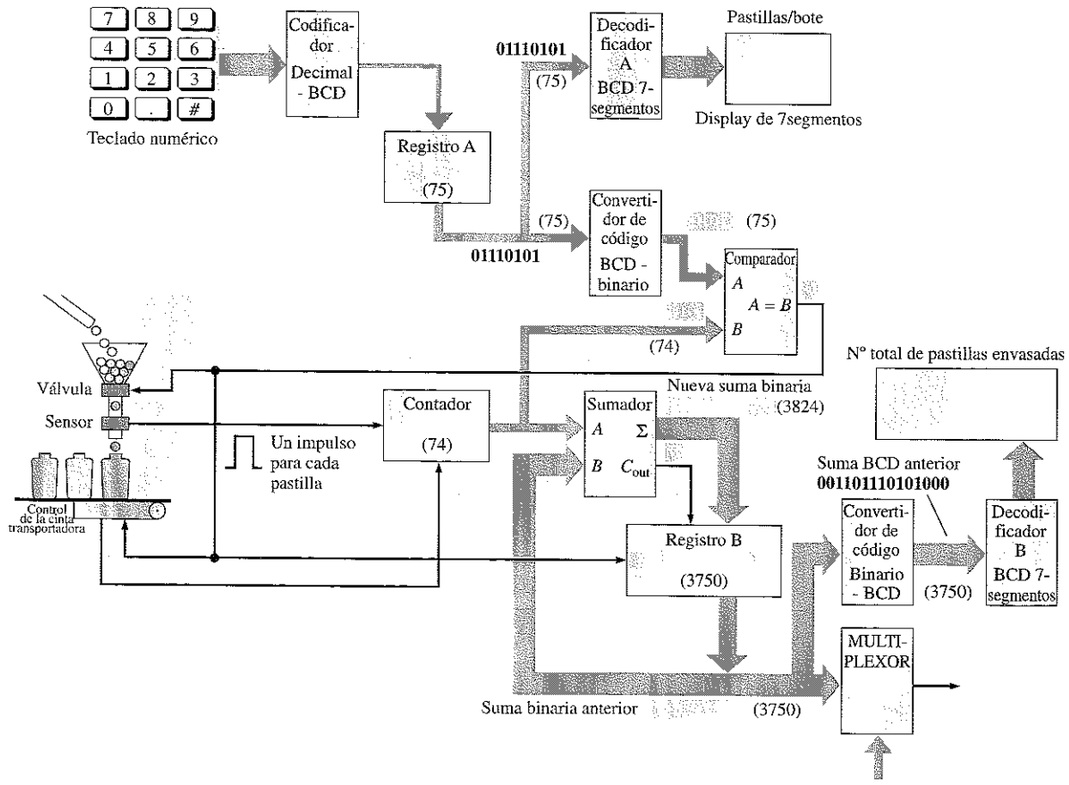


FIGURA 2.6
El sistema ha contado 50 botes de pastillas y está trabajando con el bote siguiente.

El sistema en proceso. Ahora supongamos que el sistema ha contado un total de 50 botes (3750 pastillas), y se encuentra en el proceso de contar pastillas para el bote cin-

cuenta y uno. En realidad, como indica el contenido del contador de la Figura 2.6, ha contado 74 pastillas. Se observa que el registro B mantiene la suma binaria anterior (3750), que se ha pasado a BCD y luego a formato de 7 segmentos para el display. El registro B se actualiza cada vez que se llena un bote, pero no mientras se está llenando.

La salida del comparador está a nivel BAJO (0), ya que las entradas *A* y *B* no son iguales en este instante. En la entrada *A* hay un 75 binario, y en la entrada *B* un 74 binario. La salida del sumador siempre indica la suma más reciente, en este caso, 3824, pero este número no se almacena en el registro B hasta que el bote está lleno.

En el estado siguiente, como muestra la Figura 2.7, el contador ha alcanzado el valor binario de 75, por lo que la salida $A = B$ del comparador pasa a estado ALTO (1), lo que da lugar a que ocurran varias cosas simultáneamente. Se detiene el flujo de pastillas, se permite que la nueva suma binaria (3825) a la salida del sumador se almacene en el registro B, y se inicia un avance de la cinta transportadora para colocar el siguiente bote en posición. El número binario almacenado en el registro B, que es ahora la suma actual, se aplica al convertidor de código BCD-7 segmentos, y en formato de 7 segmentos se actualiza el display con el valor 3825. Tan pronto como se coloca el siguiente bote, el contador se pone a cero y un nuevo ciclo comienza.

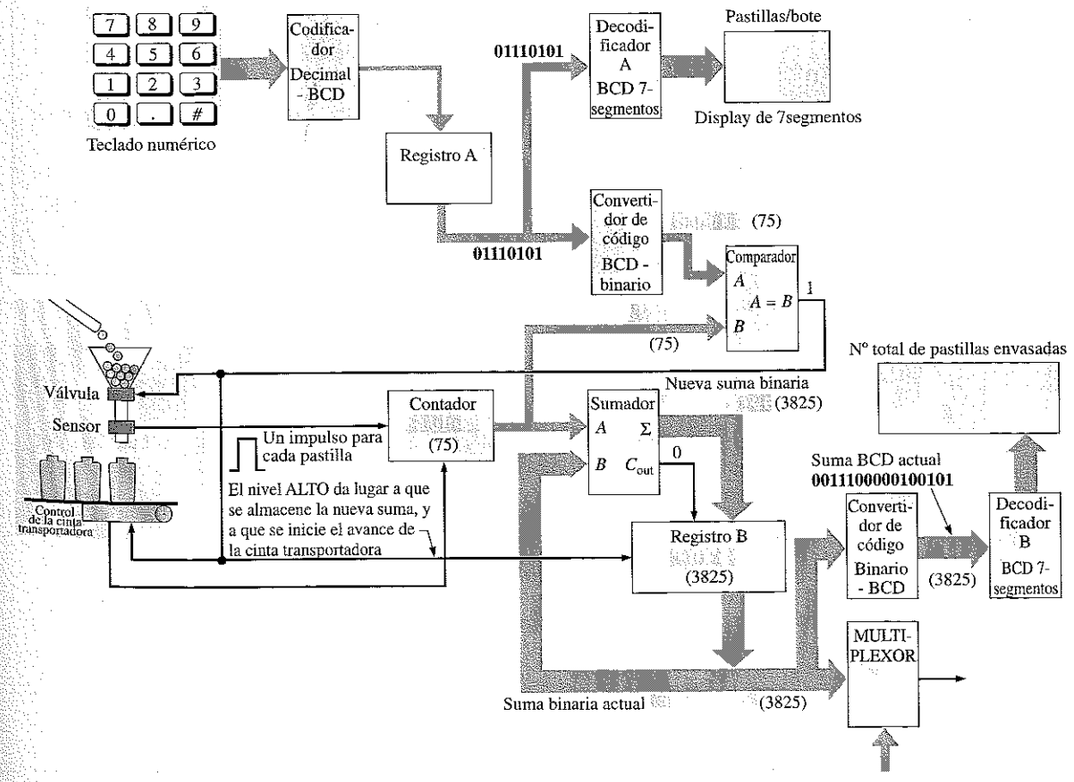


FIGURA 2.7
El sistema acaba de contar su bote de pastillas número 51.

■ PRÁCTICAS DE SISTEMAS

- ***Actividad 1: Inicialización***
  - **(a)** El sistema se configura para introducir 60 pastillas por bote. Especificar el contenido del registro A y los datos en la entrada A del comparador.
  - **(b)** ¿Cuál es el número binario máximo al que llegará el contador cuando el sistema se inicializa para 60 pastillas por bote?
- ***Actividad 2: Recuento máximo***
  - **(a)** Determinar el número máximo de botes llenos que puede contar el sistema si se ha inicializado para 75 pastillas por bote. El número de bits presentados en cada punto es el número máximo de bits.
  - **(b)** Especificar el número binario contenido en el registro B en el momento en que el último bote se ha llenado.
  - **(c)** Especificar el número BCD a la salida del convertidor de código binario-BCD en ese mismo momento.
- ***Actividad 3: Secuenciamiento***
  - **(a)** Si el sistema se inicializa para introducir 30 pastillas por bote, y en un punto dado, se han contado 523 botes, determinar el número BCD del registro A y el número binario del registro B.
  - **(b)** Si el contador contiene el número binario 00010110, determinar la salida binaria del sumador para la condición del apartado (a).
  - **(c)** ¿Cuántas pastillas en total se han contado, realmente, en el apartado (b)?
  - **(d)** ¿Cuál es el número actual en el display de totales, y cuál será el siguiente número que aparecerá en él?
- ***Actividad 4: Hexadecimal***
  - **(a)** Expresar en hexadecimal los números binarios de la Figura 2.5.
  - **(b)** Expresar en hexadecimal los números binarios de la Figura 2.6.
  - **(c)** Expresar en hexadecimal los números binarios de la Figura 2.7.
  - **(d)** Si el contador tiene el valor $4C_{16}$, y se cuentan nueve pastillas más, ¿cuál será el valor del contador en hexadecimal?
  - **(e)** Si el display, que muestra el total de pastillas en la Figura 2.6, fuera un display hexadecimal, ¿qué valor mostraría?

REVISIÓN DE LA SECCIÓN 2.12

1. En el sistema tratado en esta sección, ¿cuándo pasa la salida del comparador a nivel ALTO (H)?
2. ¿Bajo qué condiciones cambia el display que presenta el número de pastillas por bote?
3. ¿Bajo qué condiciones cambia el display que presenta el número total de pastillas?

■ RESUMEN

- Un número binario es un grupo de bits con peso en el que el peso de cada número entero es una potencia positiva de dos, y el peso de cada dígito fraccionario es una potencia negativa de dos. Los pesos de los números enteros aumentan de derecha a izquierda, del bit menos significativo al más significativo.
- Un número binario puede convertirse a número decimal sumando los valores decimales de los pesos de todos los 1s del número binario.
- Un número entero decimal puede convertirse a binario utilizando la suma de pesos o el método de la división sucesiva por 2.
- Una fracción decimal puede convertirse a binario utilizando la suma de pesos o el método de multiplicación sucesiva por 2.
- Las reglas básicas de la suma binaria son las siguientes:

$$0 + 0 = 0$$
$$1 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10$$

- Las reglas básicas de la resta binaria son las siguientes:

$$0 - 0 = 0$$
$$1 - 1 = 0$$
$$1 - 0 = 1$$
$$10 - 1 = 1$$

- El complemento a 1 de un número binario se obtiene cambiando los 1s por 0s, y los 0s por 1s.
- El complemento a 2 de un número binario puede obtenerse sumando 1 al complemento a 1.
- La resta binaria puede realizarse mediante sumas, utilizando los métodos de complemento a 1 y complemento a 2.
- Un número binario positivo se representa mediante un bit de signo 0.
- Un número binario negativo se representa mediante un bit de signo 1.
- Para las operaciones aritméticas, los números binarios negativos se representan en complemento a 1 o en complemento a 2.
- En una operación de suma, se puede producir un desbordamiento cuando ambos números son positivos o negativos. Un bit de signo incorrecto en la suma indica que se ha producido un desbordamiento.
- El sistema de numeración hexadecimal está formado por 16 dígitos y caracteres, de 0 hasta 9 y de A hasta F.
- Un dígito hexadecimal se representa mediante un número binario de cuatro bits, y su principal utilidad es simplificar los modelos binarios y hacerlos más fáciles de leer.
- Un número decimal puede convertirse a hexadecimal por el método de la división sucesiva por 16.
- El sistema de numeración octal se forma con ocho dígitos, de 0 hasta 7.
- Un número decimal puede convertirse a octal utilizando el método de la división sucesiva por 8.
- La conversión octal binario se realiza reemplazando cada dígito octal por su equivalente binario de tres bits. Para la conversión binario-octal se realiza el mismo proceso a la inversa.
- Un número decimal se convierte a BCD reemplazando cada dígito decimal por el apropiado código binario de cuatro bits.
- El código ASCII es un código alfanumérico de siete bits, que se utiliza ampliamente en sistemas de computadora para las entradas y salidas de información.

■ PALABRAS CLAVE

Estos términos también se encuentran en el glosario del final del libro.

Acarreo Dígito generado cuando la suma de dos dígitos binarios excede de 1.

ACSII American Standard Code for Information Interchange, código estándar americano para el intercambio de información; el código alfanumérico más utilizado.

Alfanumérico Que contiene números, letras y otros caracteres.

BCD Binary Coded Decimal, código decimal binario, código digital en el que cada dígito decimal, de 0 a 9, se representa mediante un grupo de cuatro bits.

Bit de signo El bit más a la izquierda de un número binario que designa si el número es positivo (0) o negativo (1).

Bit más significativo (MSB, Most Significant Bit) El bit más a la izquierda de un número entero o código binario.

Bit menos significativo (LSB, Least Significant Bit) Generalmente, el bit de más a la derecha de un número entero o código binario.

Byte Grupo de ocho bits.

Código Gray Código digital sin pesos caracterizado por el cambio de un único bit entre números codificados adyacentes en una secuencia.

Complemento El inverso u opuesto de un número.

Decimal Describe un sistema de numeración en base diez.

Desbordamiento La condición que se produce cuando el número de bits en una suma excede el número de bits de cada uno de los números que se suman.

Dígito Símbolo utilizado para expresar una cantidad.

Entero. Un número entero, sin parte fraccionaria.

Exponente La parte de un número en coma flotante que representa el número de posiciones que se desplaza la coma decimal (o coma binaria).

Hexadecimal Describe un sistema de numeración en base 16.

Mantisa La magnitud de un número en coma flotante.

Número en coma flotante Representación de un número basada en la notación científica, en la que el número consta de un exponente, una mantisa y un signo.

Octal Describe un sistema de numeración en base ocho.

Paridad En relación a los códigos binarios, tener un número par o impar de 1s en un grupo de código.

Peso El valor de un dígito en un número en función de su posición dentro del número.

■ AUTOTEST

1. $2 \times 10^1 + 8 \times 10^0$ es igual a
(a) 10 **(b)** 280 **(c)** 2,8 **(d)** 28

2. El número binario 1101 es igual al número decimal
(a) 13 **(b)** 49 **(c)** 11 **(d)** 3

3. El número binario 11011101 es igual al número decimal
(a) 121 **(b)** 221 **(c)** 441 **(d)** 256

4. El número decimal 17 es igual al número binario
(a) 10010 **(b)** 11000 **(c)** 10001 **(d)** 01001

5. El número decimal 175 es igual al número binario
(a) 11001111 **(b)** 10101110 **(c)** 10101111 **(d)** 11101111

6. La suma de 11010 + 01111 es igual a
(a) 101001 **(b)** 101010 **(c)** 110101 **(d)** 101000

7. La diferencia de 110 − 010 es igual a
(a) 001 **(b)** 010 **(c)** 101 **(d)** 100

8. El complemento a 1 de 10111001 es
(a) 01000111 **(b)** 01000110 **(c)** 11000110 **(d)** 10101010

9. El complemento a 2 de 11001000 es
(a) 00110111 **(b)** 00110001 **(c)** 01001000 **(d)** 00111000

10. El número decimal +122 expresado en complemento a 2 es
(a) 01111010 **(b)** 11111010 **(c)** 01000101 **(d)** 10000101

11. El número decimal −34 expresado en complemento a 2 es
(a) 01011110 **(b)** 10100010 **(c)** 11011110 **(d)** 01011101

12. Un número binario en coma flotante de simple precisión tiene un total de
(a) 8 bits **(b)** 16 bits **(c)** 24 bits **(d)** 32 bits

13. En el sistema de complemento a 2, el número binario 10010011 es igual al número decimal
(a) −19 **(b)** +109 **(c)** + 91 **(d)** −109

14. El número binario 10110011100101010100001 escrito en octal es
(a) 5471230_8 **(b)** 5471241_8 **(c)** 2634521_8 **(d)** 23162501_8

15. El número binario 1000110101000110111 se puede escribir en hexadecimal como
(a) $AD467_{16}$ **(b)** $8C46F_{16}$ **(c)** $8D46F_{16}$ **(d)** $AE46F_{16}$

16. El número binario correspondiente a $F7A9_{16}$ es
(a) 1111011110101001 **(b)** 1110111110101001
(c) 1111111010110001 **(d)** 1111011010101001

17. El número BCD correspondiente al decimal 473 es
(a) 111011010 **(b)** 110001110011
(c) 010001110011 **(d)** 010011110011

18. Utilizando la Tabla 2.7, indicar a qué código ASCII corresponde la palabra STOP
(a) 10100111010100100111111010000 **(b)** 10100101001100100111010100000
(c) 10010101101101100111010100001 **(d)** 10100111010100100111011100100

■ PROBLEMAS

SECCIÓN 2.1 Números decimales

1. ¿Cuál es el peso del dígito 6 en cada uno de los siguientes números decimales?
(a) 1.386 **(b)** 54.692 **(c)** 671.920

2. Expresar cada uno de los siguientes números decimales como potencias de diez.
(a) 10 **(b)** 100 **(c)** 10.000 **(d)** 1.000.000

3. Hallar el valor de cada dígito en los siguientes números decimales:
(a) 471 **(b)** 9.356 **(c)** 125.000

4. ¿Hasta qué número puede contar con cuatro dígitos decimales?

SECCIÓN 2.2 Números binarios

5. Convertir a decimal los siguientes números binarios:
(a) 11 **(b)** 100 **(c)** 111 **(d)** 1000
(e) 1001 **(f)** 1100 **(g)** 1011 **(h)** 1111

6. Convertir a decimal los siguientes números binarios:
(a) 1110 **(b)** 1010 **(c)** 11100 **(d)** 10000
(e) 10101 **(f)** 11101 **(g)** 10111 **(h)** 11111

7. Convertir a decimal los siguientes números binarios:
(a) 110011,11 **(b)** 101010,01 **(c)** 1000001,111 **(d)** 1111000,101
(e) 1011100,10101 **(f)** 1110001,0001 **(g)** 1011010,1010 **(h)** 1111111,11111

8. ¿Cuál es el mayor número decimal que se puede representar con cada una de las siguientes cantidades de dígitos binarios (bits)?
(a) dos **(b)** tres **(c)** cuatro **(d)** cinco **(e)** seis
(f) siete **(g)** ocho **(h)** nueve **(i)** diez **(j)** once

9. ¿Cuántos bits se requieren para representar los siguientes números decimales?
(a) 17 **(b)** 35 **(c)** 49 **(d)** 68
(e) 81 **(f)** 114 **(g)** 132 **(h)** 205

10. Generar la secuencia binaria para las secuencias decimales:
(a) de 0 a 7 **(b)** de 8 a 15 **(c)** de 16 a 31
(d) de 32 a 63 **(e)** de 64 a 75

SECCIÓN 2.3 Conversión decimal-binario

11. Convertir a binario cada uno de los números decimales indicados utilizando el método de la suma de pesos.
(a) 10 **(b)** 17 **(c)** 24 **(d)** 48
(e) 61 **(f)** 93 **(g)** 125 **(h)** 186

12. Convertir a binario cada uno de los números decimales fraccionarios indicados utilizando el método de la suma de pesos.
(a) 0,32 **(b)** 0,246 **(c)** 0,0981

13. Convertir a binario cada uno de los números decimales indicados utilizando la división sucesiva por 2.
(a) 15 **(b)** 21 **(c)** 28 **(d)** 34
(e) 40 **(f)** 59 **(g)** 65 **(h)** 73

14. Convertir a binario cada uno de los números decimales fraccionarios indicados utilizando la multiplicación sucesiva por 2.
(a) 0,98 **(b)** 0,347 **(c)** 0,9028

SECCIÓN 2.4 Aritmética binaria

15. Sumar los números binarios:
(a) 11 + 01 **(b)** 10 + 10 **(c)** 101 + 11
(d) 111 + 110 **(e)** 1001 + 101 **(f)** 1101 + 1011

16. Realizar la sustracción directa de los siguientes números binarios:
(a) 11 - 1 **(b)** 101 - 100 **(c)** 110 - 101
(d) 1110 - 11 **(e)** 1100 - 1001 **(f)** 11010 - 10111

17. Realizar las siguientes multiplicaciones binarias:
(a) 11 x 11 **(b)** 100 x 10 **(c)** 111 x 101
(d) 1001 x 110 **(e)** 1101 x 1101 **(f)** 1110 x 1101

18. Dividir los números binarios siguientes:
(a) 100 ÷ 10 (b) 1001 ÷ 11 (c) 1100 ÷ 100

SECCIÓN 2.5 Complemento a 1 y complemento a 2 de los números binarios

19. Determinar el complemento a 1 de los números binarios:
(a) 101 **(b)** 110 **(c)** 1010
(d) 11010111 **(e)** 1110101 **(f)** 00001

20. Determinar el complemento a 2 de los números binarios siguientes utilizando cualquier método:
(a) 10 **(b)** 111 **(c)** 1001 **(d)** 1101
(e) 11100 **(f)** 10011 **(g)** 10110000 **(h)** 00111101

SECCIÓN 2.6 Números con signo

21. Expresar en formato binario de 8 bits signo-magnitud los números decimales:
(a) +29 **(b)** –85 **(c)** +100 **(d)** –123

22. Expresar cada número decimal como un número de 8 bits en el sistema de complemento a 1:
(a) –34 **(b)** +57 **(c)** –99 **(d)** +115

23. Expresar cada número decimal como un número de 8 bits en el sistema de complemento a 2:
(a) +12 **(b)** –68 **(c)** +101 **(d)** –125

24. Determinar el valor decimal correspondiente a cada número binario con signo en formato signo-magnitud:
(a) 10011001 **(b)** 01110100 **(c)** 10111111

25. Determinar el valor decimal correspondiente a cada número binario con signo en formato complemento a 1:
(a) 10011001 **(b)** 01110100 **(c)** 10111111

26. Determinar el valor decimal correspondiente a cada número binario con signo en formato complemento a 2:
(a) 10011001 **(b)** 01110100 **(c)** 10111111

27. Expresar cada uno los siguientes números binarios signo-magnitud en formato de coma flotante de simple precisión:
(a) 0111110000101011 **(b)** 100110000011000

28. Determinar los valores de los siguientes números en coma flotante de simple precisión:
(a) 1 10000001 01001001110001000000000
(b) 0 11001100 10000111110100100000000

SECCIÓN 2.7 Operaciones aritméticas de números con signo

29. Convertir a binario cada pareja de números decimales y sumarlos utilizando el sistema de complemento a 2:
(a) 33 y 15 **(b)** 56 y −27 **(c)** −46 y 25 **(d)** −110 y −84

30. Realizar las siguientes sumas utilizando el sistema de complemento a 2
(a) 00010110 + 00110011 **(b)** 01110000 + 10101111

31. Realizar las siguientes sumas utilizando el sistema de complemento a 2
(a) 10001100 + 00111001 **(b)** 11011001 + 11100111

32. Realizar las siguientes restas utilizando el sistema de complemento a 2
(a) 00110011 − 00010000 **(b)** 01100101 − 11101000

33. Multiplicar 01101010 por 11110001 utilizando el sistema de complemento a 2.

34. Dividir 01000100 entre 00011001 utilizando el sistema de complemento a 2.

SECCIÓN 2.8 Números hexadecimales

35. Convertir a binario los números hexadecimales:
(a) 38_{16} **(b)** 59_{16} **(c)** $A14_{16}$ **(d)** $5C8_{16}$
(e) 4100_{16} **(f)** $FB17_{16}$ **(g)** $8A9D_{16}$

36. Convertir a hexadecimal los números binarios:
(a) 1110 **(b)** 10 **(c)** 10111
(d) 10100110 **(e)** 1111110000 **(f)** 100110000010

37. Convertir a decimal los números hexadecimales:
(a) 23_{16} **(b)** 92_{16} **(c)** $1A_{16}$ **(d)** $8D_{16}$
(e) $F3_{16}$ **(f)** EB_{16} **(g)** $5C2_{16}$ **(h)** 700_{16}

38. Convertir a hexadecimal los números decimales:
(a) 8 **(b)** 14 **(c)** 33 **(d)** 52
(e) 284 **(f)** 2890 **(g)** 4019 **(h)** 6500

39. Realizar las siguientes sumas:
(a) $37_{16} + 29_{16}$ **(b)** $A0_{16} + 6B_{16}$ **(c)** $FF_{16} + BB_{16}$

40. Realizar las siguientes sustracciones:
(a) $51_{16} - 40_{16}$ **(b)** $C8_{16} - 3A_{16}$ **(c)** $FD_{16} - 88_{16}$

SECCIÓN 2.9 Números octales

41. Convertir a decimal los números octales:
(a) 12_8 **(b)** 27_8 **(c)** 56_8 **(d)** 64_8 **(e)** 103_8
(f) 557_8 **(g)** 163_8 **(h)** 1024_8 **(i)** 7765_8

42. Convertir a octal los números decimales siguientes utilizando la división sucesiva por 8:
(a) 15 **(b)** 27 **(c)** 46 **(d)** 70
(e) 100 **(f)** 142 **(g)** 219 **(h)** 435

43. Convertir a binario los números octales:
(a) 13_8 **(b)** 57_8 **(c)** 101_8 **(d)** 321_8 **(e)** 540_8
(f) 4653_8 **(g)** 13271_8 **(h)** 45600_8 **(i)** 100213_8

44. Convertir a octal los números binarios:
(a) 111 **(b)** 10 **(c)** 110111 **(d)** 101010 **(e)** 1100
(f) 1011110 **(g)** 101100011001 **(h)** 10110000011 **(i)** 111111101111000

SECCIÓN 2.10 Código decimal binario (BCD)

45. Convertir a BCD 8421 los siguientes números decimales:
(a) 10 **(b)** 13 **(c)** 18 **(d)** 21
(e) 25 **(f)** 36 **(g)** 44 **(h)** 57
(i) 69 **(j)** 98 **(k)** 125 **(l)** 156

46. Convertir los números decimales del problema 45 a binario normal, y comparar el número de bits requerido con los que se requieren para BCD.

47. Convertir a BCD los siguientes números decimales:
(a) 104 **(b)** 128 **(c)** 132 **(d)** 150
(e) 186 **(f)** 210 **(g)** 359 **(h)** 547 **(i)** 1051

48. Convertir a decimal los números en BCD:
(a) 0001 **(b)** 0110 **(c)** 1001 **(d)** 00011000 **(e)** 00011001
(f) 00110010 **(g)** 01000101 **(h)** 10011000 **(i)** 100001110000

49. Convertir a decimal los números en BCD:
(a) 10000000 **(b)** 001000110111
(c) 001101000110 **(d)** 010000100001
(e) 011101010100 **(f)** 100000000000
(g) 100101111000 **(h)** 0001011010000011
(i) 1001000000011000 **(j)** 0110011001100111

50. Sumar los siguientes números en BCD:
(a) 0010 + 0001 **(b)** 0101 + 0011
(c) 0111 + 0010 **(d)** 1000 + 0001
(e) 00011000 + 00010001 **(f)** 01100100 + 00110011
(g) 01000000 + 01000111 **(h)** 10000101 + 00010011

51. Sumar los siguientes números en BCD:
(a) 1000 + 0110 **(b)** 0111 + 0101
(c) 1001 + 1000 **(d)** 1001 + 0111
(e) 00100101 + 00100111 **(f)** 01010001 + 01011000
(g) 10011000 + 10010111
(h) 010101100001 + 011100001000

52. Convertir a BCD cada pareja de números decimales y sumarlos:
(a) 4 + 3 **(b)** 5 + 2 **(c)** 6 + 4 **(d)** 17 + 12
(e) 28 + 23 **(f)** 65 + 58 **(g)** 113 + 101 **(h)** 295 + 157

SECCIÓN 2.11 Códigos digitales y paridad

53. En una determinada aplicación se producen ciclos de una secuencia binaria de 4 bits de 1111 a 0000 de forma periódica. Existen cuatro variaciones de bit, y debido a retrasos del circuito, estas variaciones pueden no producirse en el mismo instante. Por ejemplo, si el LSB cambia el primero, entonces durante la transición de 1111 a 0000 aparecerá el número 1110, y puede ser mal interpretado por el sistema. Ilustrar cómo resuelve este problema el código Gray.

54. Convertir a código Gray los números binarios:
(a) 11011 **(b)** 1001010 **(c)** 1111011101110

55. Convertir a binario los números en código Gray:
(a) 1010 **(b)** 00010 **(c)** 11000010001

56. Convertir a código ASCII cada uno de los siguientes números decimales. Utilice la Tabla 2.7.
(a) 1 **(b)** 3 **(c)** 6 **(d)** 10 **(e)** 18
(f) 29 **(g)** 56 **(h)** 75 **(i)** 107

57. Determinar el carácter de cada uno de los siguientes códigos ASCII. Utilice la Tabla 2.7.
(a) 0011000 **(b)** 1001010 **(c)** 0111101
(d) 0100011 **(e)** 0111110 **(f)** 1000010

58. Decodificar el siguiente mensaje codificado en ASCII:
1001000 1100101 1101100 1101100 1101111 0101110
0100000 1001000 1101111 1110111 0100000 1100001
1110010 1100101 0100000 1111001 1101111 1110101
0111111

59. Escribir en hexadecimal el mensaje del problema 58.

60. Convertir a código ASCII la siguiente instrucción de programa para una computadora:

```
30 INPUT A, B
```

61. Determinar cuáles de los siguientes códigos con paridad par son erróneos:
(a) 100110010 **(b)** 011101010 **(c)** 10111111010001010

62. Determinar cuáles de los siguientes códigos con paridad impar son erróneos:
(a) 11110110 **(b)** 00110001 **(c)** 01010101010101010

63. Añadir el bit de paridad par apropiado a los siguientes bytes de datos:
(a) 10100100 **(b)** 00001001 **(c)** 11111110

SECCIÓN 2.12 Aplicación a los sistemas digitales

Aplicación a sistemas

64. En el sistema de control y recuento de pastillas, ¿qué limita el número máximo de pastillas por bote que se puede preseleccionar en el sistema?

65. Supongamos que el sistema se inicializa para 25 pastillas por bote. Determinar las cantidades binarias en cada una de las siguientes posiciones dentro del sistema, después de haber introducido 15 pastillas en el bote número 1000:
(a) En el registro A **(b)** En la entrada A del comparador
(c) En el contador **(d)** En el registro B
(e) En la salida del sumador.

66. Determinar el número máximo de pastillas que en total puede contar el sistema. Explicar qué cambios se requerirían para aumentar este número.

■ RESPUESTAS A LAS REVISIONES DE CADA SECCIÓN

SECCIÓN 2.1

1. **(a)** 1370 : 10 **(b)** 6725: 100 **(c)** 7051: 1000 **(d)** 58,72: 0,1

2. **(a)** 51 = (5 x10) + (1 x 1)
(b) 137 = (1 x 100) + (3 x 10) + (7 x 1)
(c) 1492 = (1 x 1000) + (4 x100) + (9 x 10) + (2 x 1)
(d) 106,58 = (1 x 100) + (0 x 10) + (6 x 1) + (5 x 0,1) + (8 x 0,01)

SECCIÓN 2.2

1. $2^8-1 = 255$

2. El peso es 16

3. 10111101,011 = 189,375

SECCIÓN 2.3

1. **(a)** 23 = 10111 **(b)** 57 = 111001 **(c)** 45,5 = 101101,1

2. **(a)** 14 = 1110 **(b)** 21 = 10101 **(c)** 0,375 = 0,011

SECCIÓN 2.4

1. **(a)** 1101 + 1010 = 10111 **(b)** 10111 + 01101 = 100100

2. **(a)** 1101 – 0100 = 1001 **(b)** 1001 – 0111 = 0010

3. **(a)** 110 x 111 = 101010 **(b)** 1100 ÷ 011 = 100

SECCIÓN 2.5

1. **(a)** Complemento a 1 de 00011010 = 11100101
(b) Complemento a 1 de 11110111 = 00001000
(c) Complemento a 1 de 10001101 = 01110010

2. **(a)** Complemento a 2 de 00010110 = 11101010
(b) Complemento a 2 de 11111100 = 00000100
(c) Complemento a 2 de 10010001 = 01101111

SECCIÓN 2.6

1. Signo-magnitud de +9 = 00001001

2. Complemento a 1: –33 = 11011110

3. Complemento a 2: –46 = 11010010

4. Exponente y mantisa

SECCIÓN 2.7

1. Casos de la suma: el número positivo es el mayor, el número negativo es el mayor (en valor absoluto), ambos son positivos y ambos son negativos.

2. 00100001 + 10111100 = 11011101

3. 01110111 – 00110010 = 01000101

4. El signo del producto es positivo.

5. 00000101 x 01111111 = 01001111011

6. El signo del cociente es negativo

7. 00110000 ÷ 00001100 = 00000100

SECCIÓN 2.8

1. **(a)** $10110011 = B3_{16}$ **(b)** $110011101000 = CE8_{16}$

2. **(a)** $57_{16} = 01010111$ **(b)** $3A5_{16} = 001110100101$
(c) $F8OB_{16} = 1111100000001011$

3. $9B30_{16} = 39.728_{10}$ **4.** $573_{10} = 23D_{16}$

5. **(a)** $18_{16} + 34_{16} = 4C_{16}$ **(b)** $3F_{16} + 2A_{16} = 69_{16}$

6. **(a)** $75_{16} - 21_{16} = 54_{16}$ **(b)** $94_{16} - 5C_{16} = 38_{16}$

SECCIÓN 2.9

1. **(a)** $73_8 = 59_{10}$ **(b)** $125_8 = 85_{10}$

2. **(a)** $98_{10} = 142_8$ **(b)** $163_{10} = 243_8$

3. **(a)** $46_8 = 100110$ **(b)** $723_8 = 111010011$ **(c)** $5624_8 = 101110010100$

4. **(a)** $110101111 = 657_8$ **(b)** $1001100010 = 1142_8$ **(c)** $10111111001 = 2771_8$

SECCIÓN 2.10

1. **(a)** 0010: 2 **(b)** 1000: 8 **(c)** 0001: 1 **(d)** 0100: 4

2. **(a)** $6_{10} = 0110$ **(b)** $15_{10} = 00010101$ **(c)** $273_{10} = 001001110011$
(d) $849_{10} = 100001001001$

3. **(a)** $10001001 = 89_{10}$ **(b)** $001001111000 = 278_{10}$ **(c)** $000101010111 = 157_{10}$

4. Una suma de 4 bits no es válida cuando es mayor que 9_{10}.

SECCIÓN 2.11

1. **(a)** $1100_2 = 1010$ Gray **(b)** $1010_2 = 1111$ Gray **(c)** $11010_2 = 10111$ Gray

2. **(a)** 1000 Gray = 1111_2 **(b)** 1010 Gray = 1100_2 **(c)** 11101 Gray = 10110_2

3. **(a)** K: $1001011 \rightarrow 4B_{16}$ **(b)** r: $1110010 \rightarrow 72_{16}$
(c) $: $0100100 \rightarrow 24_{16}$ **(d)** +: $0101011 \rightarrow 2B_{16}$

4. **(a)** *0*1001011 **(b)** *0*1110010 **(c)** *0*0100100 **(d)** *0*0101011

SECCIÓN 2.12

1. La salida del comparador pasa a nivel ALTO (H) cuando el número actual de pastillas contadas en un bote es igual al número preseleccionado.

2. El número de pastillas/bote en el display cambia cuando se introduce un nuevo número a través del teclado numérico.

3. El número total de pastillas en el display de totales cambia cada vez que se llena un bote.

■ RESPUESTAS A LOS EJERCICIOS RELACIONADOS DE LOS EJEMPLOS

2.1 9 tiene un valor de 900, 3 tiene un valor de 30 y 9 tiene un valor de 9.

2.2 6 tiene un valor de 60, 7 tiene un valor de 7, 9 tiene un valor de 9/10 (0,9), 2 tiene un valor de 2/100 (0,02), 4 tiene un valor de 4/1000 (0,004).

2.3 10010001 = 128 + 16 + 1 = 145

2.4 10,111 = 2 + 0,5 + 0,25 + 0,125 = 2,875

2.5 125 = 64 + 32 + 16 + 8 + 4 + 1 = 1111101

2.6 39 = 100111

2.7 1111 + 1100 = 11011

2.8 111 − 100 = 011

2.9 110 − 101 = 001

2.10 1101 × 1010 = 10000010

2.11 1100 ÷ 100 = 11

2.12 00110101

2.13 01000000

2.14 Véase la Tabla 2.11.

TABLA 2.11

	Signo-magnitud	Complemento a 1	Complemento a 2
+19	00010011	00010011	00010011
−19	10010011	11101100	11101101

2.15 $01110111 = +119_{10}$

2.16 $11101011 = -20_{10}$

2.17 $11010111 = -41_{10}$

2.18 110000100010100110000000000

2.19 01010101

2.20 00010001

2.21 1001000110

2.22 (83)(−59) = −4897 (10110011011111 en complemento a 2)

2.23 100 ÷ 25 = 4 (0100)

2.24 $4F79C_{16}$

2.25 0110101111010011_2

2.26 $6BD_{16} = 011010111101 = 2^{10} + 2^9 + 2^7 + 2^5 + 2^4 + 2^3 + 2^2 + 2^0$
$= 1024 + 512 + 128 + 32 + 16 + 8 + 4 + 1 = 1725_{10}$

2.27 $60A_{16} = (6 \times 256) + (0 \times 16) + (10 \times 1) = 1546_{10}$

2.28 $2591_{10} = A1F_{16}$

2.29 $4C_{16} + 3A_{16} = 86_{16}$

2.30 $BCD_{16} - 173_{16} = A5A_{16}$

2.31 **(a)** $001011_2 = 11_{10} = 13_8$ **(b)** $010101_2 = 21_{10} = 25_8$
(c) $001100000_2 = 96_{10} = 140_8$ **(d)** $111101010110_2 = 3926_{10} = 7526_8$

2.32 1250762_8

2.33 1001011001110011

2.34 82.276_{10}

2.35 1001100101101000

2.36 10000010

2.37 **(a)** 111011 (Gray) **(b)** 111010_2

2.38 La secuencia de códigos para 80 INPUT Y es $38_{16}30_{16}20_{16}49_{16}4E_{16}50_{16}55_{16}54_{16}20_{16}59_{16}$

2.39 01001011

2.40 Sí

■ RESPUESTAS AL AUTOTEST

1. (d) **2.** (a) **3.** (b) **4.** (c) **5.** (c) **6.** (a) **7.** (d) **8.** (b)
9. (d) **10.** (a) **11.** (c) **12.** (d) **13.** (d) **14.** (b) **15.** (c) **16.** (a)
17. (c) **18.** (a)

3

PUERTAS LÓGICAS

3.1 El inversor (puerta NOT)
3.2 La puerta AND
3.3 La puerta OR
3.4 La puerta NAND
3.5 La puerta NOR
3.6 Puertas OR-exclusiva y NOR-exclusiva
3.7 Puertas lógicas integradas
3.8 Localización de averías
3.9 Aplicación a los sistemas digitales

Simulación por computadora de circuitos digitales:
El tutorial de Electronics Workbench (EWB) se encuentra en la dirección http://www.prenhall.com/floyd

■ OBJETIVOS DEL CAPÍTULO

- ❑ Describir el funcionamiento del inversor y de las puertas AND y OR.
- ❑ Describir el funcionamiento de las puertas NAND y NOR.
- ❑ Expresar las operaciones de las puertas NOT, AND, OR, NAND y NOR mediante el álgebra de Boole.
- ❑ Describir el funcionamiento de las puertas OR-exclusiva y NOR-exclusiva.
- ❑ Reconocer y utilizar los símbolos distintivos y los símbolos rectangulares de las puertas lógicas según el estándar ANSI/IEEE 91-1984.
- ❑ Elaborar los diagramas de tiempos que muestran las relaciones de tiempo de las entradas y las salidas de las diferentes puertas lógicas.
- ❑ Establecer las comparaciones básicas entre las principales tecnologías de circuitos integrados: TTL y CMOS.
- ❑ Explicar las diferencias entre las series de las familias TTL y CMOS.
- ❑ Definir, para las puertas lógicas, los siguientes parámetros: *tiempo de retardo de propagación, disipación de potencia, producto velocidad-potencia y fan-out.*
- ❑ Enumerar dispositivos específicos que contengan varias puertas lógicas.
- ❑ Utilizar cada puerta lógica en aplicaciones sencillas.
- ❑ Localización de averías en las puertas lógicas debidas a circuitos abiertos o cortocircuitos, utilizando el pulsador y la sonda lógica o el osciloscopio.
- ❑ Aplicar las puertas lógicas a un sistema digital.

■ INTRODUCCIÓN

Este capítulo hace énfasis en el funcionamiento lógico, las aplicaciones y la localización de averías de las puertas lógicas. Se cubre la relación entre las formas de onda de entrada y de salida de una puerta utilizando los diagramas de tiempos.

Los símbolos lógicos que se usan para representar las puertas lógicas están de acuerdo con el estándar ANSI/IEEE 91-1984. Este estándar ha sido adoptado por la industria privada, y la industria militar la utiliza para su documentación interna así como para sus publicaciones.

Puesto que en todas las aplicaciones se usan los circuitos integrados (CI), generalmente, la función lógica de un dispositivo es más importante para el técnico que los detalles de operación del circuito en el nivel de componentes en el interior del CI. Por tanto, la cobertura detallada de los dispositivos en el nivel de componente puede tratarse como un tema opcional. Para aquéllos que lo necesiten y tengan tiempo, en el Capítulo 15 se cubren las tecnologías de los circuitos integrados digitales. Parte de dicho capítulo puede verse después de éste, o en cualquier otro punto del texto que se considere apropiado. No obstante, el Capítulo 15 puede ser omitido total o parcialmente según desee el lector, ya que no afecta al resto del material cubierto en el libro.

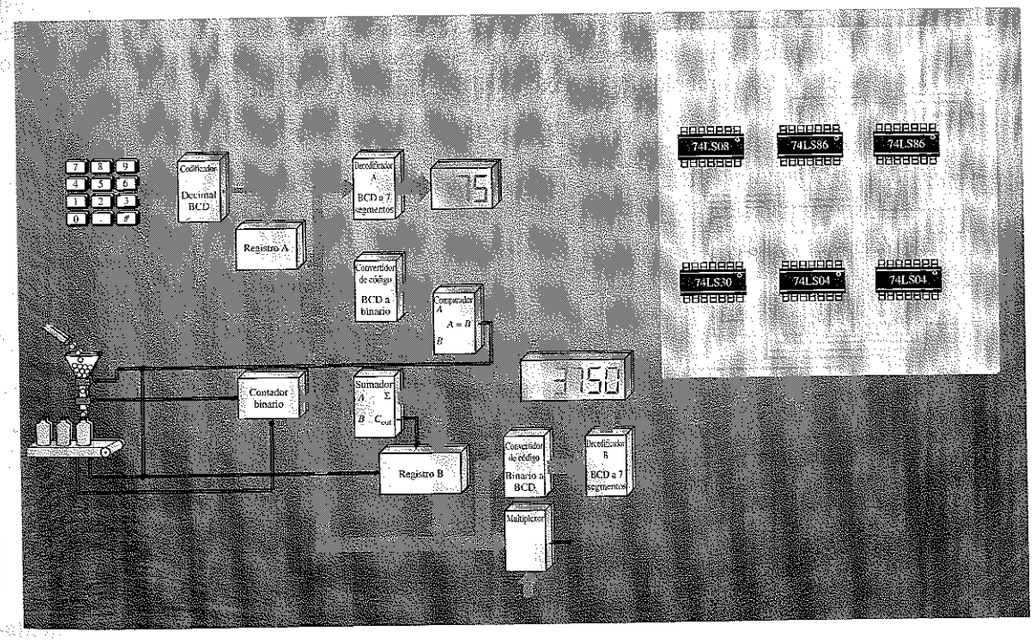
DISPOSITIVOS ESPECÍFICOS (SERIES CMOS Y TTL)

74XX00	74XX02	74XX04
74XX08	74XX10	74XX11
74XX20	74XX21	74XX27
74XX30	74XX32	74XX86
74XX133		

■ APLICACIÓN A LOS SISTEMAS DIGITALES

Esta aplicación a un sistema digital ilustra los conceptos que se enseñan en este capítulo. Las aplicaciones a los sistemas de este texto pretenden presentar sistemas "virtuales" que proporcionen cierta perspectiva acerca de cómo pueden utilizarse los circuitos lógicos de manera funcional y útil. Esto ayudará a desarrollar la capacidad de comprender el "concepto del sistema" y a ubicar el estudio de circuitos individuales en el contexto de la realidad.

Para ilustrar que las puertas lógicas son los bloques de construcción básicos de funciones más complejas, en la Sección 3.9 se implementa y analiza la función del comparador del sistema de control y recuento de pastillas que se ha introducido en los capítulos anteriores. También se examina un determinado problema de temporización del sistema, y se presenta una modificación que utiliza elementos lógicos básicos. Se localizará la avería de un mal funcionamiento del sistema especificado utilizando el pulsador y la sonda lógicos sobre la tarjeta de circuito impreso del sistema.



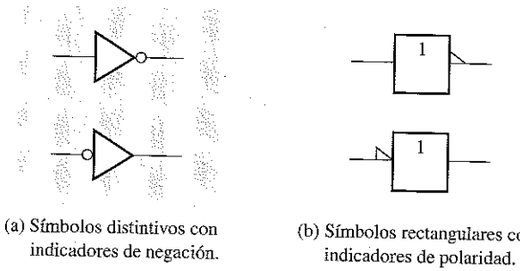
3.1 ■ EL INVERSOR

El inversor (puerta NOT) realiza la operación denominada inversión o complementación. El inversor cambia un nivel lógico al nivel opuesto. En términos de bits, cambia un 1 por un 0, y un 0 por un 1.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Identificar los indicadores de negación y polaridad. ■ Identificar un inversor tanto mediante su símbolo distintivo como por su símbolo rectangular. ■ Elaborar la tabla de verdad del inversor. ■ Describir el funcionamiento lógico de un inversor.

En la Figura 3.1 se muestran los símbolos lógicos estándar del **inversor**. La parte (a) muestra los *símbolos distintivos*, y la (b) muestra los *símbolos rectangulares*. En este texto se usan los símbolos distintivos; sin embargo, los símbolos rectangulares suelen encontrarse en las documentaciones industriales, por lo que debería familiarizarse con ellos. Los símbolos lógicos cumplen el estándar **ANSI/IEEE** 91-1984.



(a) Símbolos distintivos con indicadores de negación.

(b) Símbolos rectangulares con indicadores de polaridad.

FIGURA 3.1
Símbolos lógicos estándar de la puerta inversora (Estándar ANSI/IEEE 91-1984).

Los indicadores de negación y de polaridad

El indicador de negación es un "círculo" (○) que indica **inversión** o **complementación**, cuando aparece en la entrada o en la salida de un elemento lógico, tal como muestra la Figura 3.1(a). Generalmente, las entradas se sitúan a la izquierda del símbolo lógico, y la salida a la derecha. Cuando en la entrada hay un círculo, quiere decir que el estado activo o *verdadero* de la entrada es 0. Cuando el círculo se sitúa en la salida significa que el estado activo o verdadero de salida es 0. La ausencia de círculo en la entrada o en la salida significa que el estado activo o verdadero es 1.

El indicador de polaridad o de nivel es un "triángulo" (▷) que indica inversión cuando aparece a la entrada o a la salida de un elemento lógico, como muestra la Figura 3.1(b). Cuando se presenta a la entrada, significa que un nivel BAJO es el estado de entrada activo o verdadero. Cuando se presenta a la salida, significa que un nivel BAJO es el estado de salida activo o verdadero.

Ambos indicadores (círculo y triángulo) pueden utilizarse tanto en los símbolos distintivos como en los rectangulares. La Figura 3.1(a) indica el principal uso de los símbolos del inversor en este texto. Obsérvese que un cambio de posición del indicador de polaridad o de negación no implica un cambio en el modo de funcionamiento del inversor.

Tabla de verdad del inversor

Cuando se aplica un nivel ALTO a la entrada de un inversor, en su salida se presenta un nivel BAJO. Cuando se aplica un nivel BAJO a la entrada, en su salida se presenta un nivel ALTO. En la Tabla 3.1 se resume esta operación. Esta tabla muestra la salida para cada posible entrada en términos de niveles y bits correspondientes. Una tabla tal como ésta se llama **tabla de verdad**.

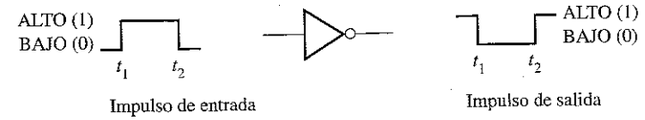
TABLA 3.1
Tabla de verdad del inversor

Entrada	Salida
BAJO (0)	ALTO (1)
ALTO (1)	BAJO (0)

Funcionamiento del inversor

La Figura 3.2 muestra la salida de un inversor para un impulso de entrada, donde t_1 y t_2 indican los puntos que corresponden a los impulsos de entrada y salida.

Cuando la entrada está a nivel BAJO, la salida está a nivel ALTO; y cuando la entrada está a nivel ALTO, la salida está a nivel BAJO, lo que da lugar a un impulso de salida invertido.



Impulso de entrada

Impulso de salida

FIGURA 3.2
Impulso de entrada en el inversor.

Diagramas de tiempos

Recuérdese del Capítulo 1 que un **diagrama de tiempos o cronograma** es básicamente *una gráfica que presenta de forma precisa las relaciones de dos o más formas de onda en función del tiempo*. Por ejemplo, la relación de tiempo del impulso de salida respecto al impulso de entrada de la Figura 3.2 puede representarse con un sencillo diagrama de tiempos, alineando los dos impulsos de modo que las ocurrencias de los flancos se presenten en los instantes de tiempo correctos. El flanco de subida del impulso de entrada y el flanco de bajada del impulso de salida se producen al mismo tiempo (idealmente). Igualmente, el flanco de bajada del impulso de entrada y el flanco de subida del impulso de salida se producen al mismo tiempo (idealmente). En la Figura 3.3 se muestra la relación de tiempos. Los diagramas de tiempos son muy útiles para ilustrar las relaciones de las señales digitales de impulsos múltiples, tal como ilustra el Ejemplo 3.1.

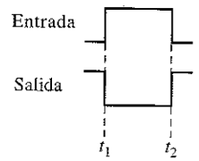


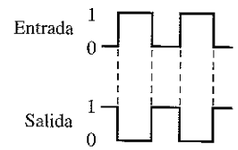
FIGURA 3.3
Diagrama de tiempos para el caso de la Figura 3.2.

EJEMPLO 3.1 Al inversor de la Figura 3.4 se le aplica una señal. Determinar la forma de onda de salida correspondiente a la entrada y dibujar el diagrama de tiempos. De acuerdo con el emplazamiento del círculo ¿cuál es el estado activo de salida?

FIGURA 3.4

Solución. La forma de onda de salida es exactamente la opuesta a la de entrada (es la entrada invertida), como se muestra en la Figura 3.5, que es el cronograma básico. El estado activo o verdadero de salida es **0**.

FIGURA 3.5



Problema relacionado. Si el inversor tiene el indicador negativo (círculo) en la entrada en lugar de en la salida, ¿cómo afecta esto al diagrama de tiempos?

Expresión lógica del inversor

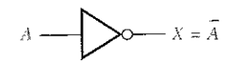
En el **álgebra booleana**, que son las matemáticas de los circuitos lógicos y que se cubrirán en el Capítulo 4, una variable se designa mediante una letra. El **complemento** de una variable se designa mediante una barra encima de la letra. Una variable puede tomar uno de dos valores, 1 o 0. Si una variable dada es 1, su complemento es 0, y viceversa.

El modo de operación de un inversor (puerta NOT) puede expresarse del siguiente modo: si la variable de entrada se designa por A y la variable de salida por X, entonces

$$X = \overline{A}$$

Esta expresión establece que la salida es el complemento de la entrada, es decir, si $A = 0$, entonces $X = 1$; y si $A = 1$, entonces $X = 0$. La Figura 3.6 ilustra esto. La variable complementada $\overline{A}$ se lee como "A negada".

FIGURA 3.6
El inversor complementa una variable de entrada.



Ejemplo de aplicación

La Figura 3.7 muestra un circuito que genera el complemento a 1 de un número binario de 8 bits. Los bits del número binario se aplican a las entradas del inversor y el complemento a 1 del número se presenta en las salidas.

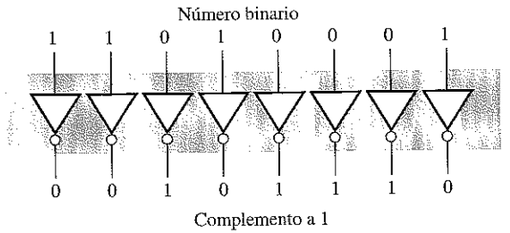


FIGURA 3.7
Ejemplo de un circuito que genera el complemento a 1 utilizando inversores.

REVISIÓN DE LA SECCIÓN 3.1

1. Cuando en la entrada de un inversor hay un 1, ¿cuál es la salida?
2. En la entrada de un inversor se requiere un impulso activo a nivel ALTO (el nivel ALTO es verdadero, y el nivel BAJO no).
   (a) Dibujar el símbolo lógico correspondiente, utilizando el indicador de negación y el símbolo distintivo para el inversor de esta aplicación.
   (b) Describir la salida cuando un impulso de subida se aplica a la entrada del inversor.

3.2 ■ LA PUERTA AND

La puerta AND es una de las puertas básicas con la que se construyen todas las funciones lógicas. Una puerta AND puede tener dos o más entradas y realiza la operación que se conoce como multiplicación lógica.

Al finalizar este capítulo, el lector deberá ser capaz de:

■ Identificar una puerta AND mediante su símbolo distintivo y su símbolo rectangular. ■ Describir la operación lógica de una puerta AND. ■ Generar la tabla de verdad de una puerta AND con cualquier número de entradas. ■ Generar el cronograma de una puerta AND para cualquier forma de onda especificada en sus entradas. ■ Escribir la expresión lógica de una puerta AND con cualquier número de entradas. ■ Analizar ejemplos de aplicaciones de la puerta AND.

El término **puerta** se usa para describir un circuito que realiza una operación lógica básica. La puerta AND tiene dos o más entradas y una única salida, como indican los

NOTAS INFORMÁTICAS
Las puertas lógicas son los bloques de construcción de las computadoras. La mayor parte de las funciones en una computadora, con la excepción de ciertos tipos de memoria, se implementan mediante puertas lógicas utilizadas a muy gran escala. Por ejemplo, un microprocesador, que es la parte principal de una computadora, consta de cientos de miles de puertas lógicas.

símbolos lógicos estándar mostrados en la Figura 3.8. En cada uno de los símbolos, las entradas se sitúan a la izquierda y la salida a la derecha. Se muestran puertas con dos entradas, pero una puerta AND puede tener cualquier número de entradas superior a éste. Aunque en los ejemplos se utilizan ambos tipos de símbolos, distintivos y rectangulares, en este libro, predominantemente, se emplea el símbolo distintivo de la Figura 3.8(a).

(a) Símbolo distintivo.

(b) Símbolo rectangular, identificado mediante el carácter AND (&).

FIGURA 3.8
Símbolos lógicos estándar de la puerta AND con dos entradas (estándar ANSI/IEEE 91-1984).

Operación lógica de la puerta AND

La **puerta AND** genera una salida a nivel ALTO *sólo* cuando *todas* las entradas están a nivel ALTO. Cuando cualquiera de las entradas está a nivel BAJO, la salida se pone a nivel BAJO. Por tanto, el propósito básico de una puerta AND es determinar cuándo ciertas condiciones de entrada son simultáneamente verdaderas, como indican todas sus entradas estando a nivel ALTO, y producir una salida a nivel ALTO, para indicar que esas condiciones son verdaderas. Las entradas de la puerta AND de dos entradas de la Figura 3.8 se designan mediante *A* y *B*, y la salida con *X*, luego podemos establecer que el funcionamiento de la puerta es el siguiente:

> **En una puerta AND de dos entradas, la salida *X* es un nivel ALTO si *A* y *B* están a nivel ALTO; y *X* es un nivel BAJO si *A* es un nivel BAJO, o si *B* es un nivel BAJO, o si *A* y *B* están a nivel BAJO.**

La Figura 3.9 ilustra una puerta AND de 2 entradas en la que se indican las cuatro posibles combinaciones de entrada y el resultado correspondiente a cada una de ellas.

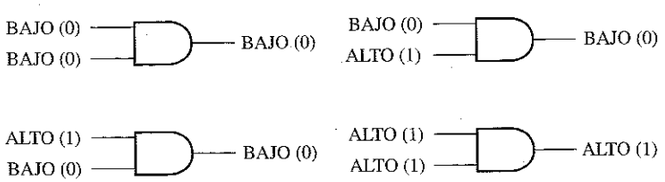


FIGURA 3.9
Todos los posibles niveles lógicos para una puerta AND de dos entradas.

Tabla de verdad de la puerta AND

La operación lógica de una puerta puede expresarse mediante una tabla de verdad, en la que se enumeran todas las combinaciones de entrada con las correspondientes salidas, como muestra la Tabla 3.2 para una puerta AND de dos entradas. La tabla de verdad puede ampliarse para cualquier número de entradas. Aunque los términos nivel ALTO y nivel BAJO dan un sentido "físico" a los estados de entrada y salida, la tabla de verdad se presenta con 1s y 0s, ya que un nivel ALTO es equivalente a un 1, y un nivel BAJO es equivalente a 0 en lógica positiva. Para cualquier puerta AND, independientemente del número de entradas, la salida es un nivel ALTO *sólo* cuando *todas* las entradas están a nivel ALTO.

TABLA 3.2
Tabla de verdad de una puerta AND de dos entradas.

Entradas		Salida
A	***B***	***X***
0	0	0
0	1	0
1	0	0
1	1	1

1 = ALTO (H), 0 = BAJO (L)

El número total de posibles combinaciones de entradas binarias a una puerta viene determinado por la siguiente fórmula:

$$N = 2^n \tag{3.1}$$

donde N son todas las posibles combinaciones, y n es el número de variables de entrada, luego

Para dos variables de entradas: $N = 2^2 = 4$ combinaciones

Para tres variables de entradas: $N = 2^3 = 8$ combinaciones

Para cuatro variables de entradas: $N = 2^4 = 16$ combinaciones

Utilizando la Ecuación 3.1 se puede determinar el número de combinaciones de bits de entrada para puertas con cualquier número de entradas. En la calculadora, pulse 2 ^ y luego introduzca el valor de n seguido de ENTER.

EJEMPLO 3.2

(a) Desarrollar la tabla de verdad de una puerta AND de 3 entradas.

(b) Determinar el número total de posibles combinaciones de entrada para una puerta AND de 4 entradas.

Solución

(a) Para una puerta AND de 3 entradas existen ocho posibles combinaciones de entrada ($2^3 = 8$). Las entradas de la tabla de verdad (Tabla 3.3) muestran las ocho posibles combinaciones de tres bits. La salida es siempre 0, excepto cuando los tres bits de entrada son 1.

TABLA 3.3

Entradas			Salida
A	*B*	*C*	*X*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

(b) $N = 2^4 = \mathbf{16}$. Para una puerta AND de 4 entradas existen 16 posibles combinaciones de bits de entrada.

Problema relacionado. Desarrollar la tabla de verdad para una puerta AND de 4 entradas.

Funcionamiento con trenes de impulsos

En la mayoría de las aplicaciones, las entradas a una puerta no son niveles estacionarios, sino que son tensiones que cambian frecuentemente entre los niveles lógicos ALTO y BAJO. Ahora vamos a ver el funcionamiento de las puertas AND con entradas que son señales digitales, teniendo en mente que una puerta AND obedece a su tabla de verdad, independientemente de que sus entradas sean niveles constantes o señales que varíen de un nivel a otro.

Al examinar el funcionamiento de una puerta AND con trenes de impulsos, nos fijaremos en los niveles de entrada para determinar el nivel de salida en cualquier instante dado. Por ejemplo, en la Figura 3.10, ambas entradas están a nivel ALTO durante el intervalo de tiempo t_1, por lo que la salida en este intervalo estará a nivel ALTO. Durante el intervalo t_2, la entrada *A* está a nivel BAJO (0) y la entrada *B* está a nivel ALTO (1), por lo que la salida se pondrá a nivel BAJO (0). De nuevo, durante el intervalo t_3, ambas entradas están a nivel ALTO (1) y, por tanto, la salida está a nivel ALTO (1). Durante el intervalo t_4, la entrada *A* está a nivel ALTO (1) y la *B* está a nivel BAJO (0), luego la salida está a nivel BAJO (0). Por último, durante el intervalo t_5, la entrada *A* está a nivel BAJO (0) y la entrada *B* está a nivel BAJO (0) y, por tanto, la salida está a nivel BAJO (0). Como ya se sabe, un diagrama de las señales de entrada y de salida en función del tiempo se llama *diagrama de tiempos* o *cronograma*.

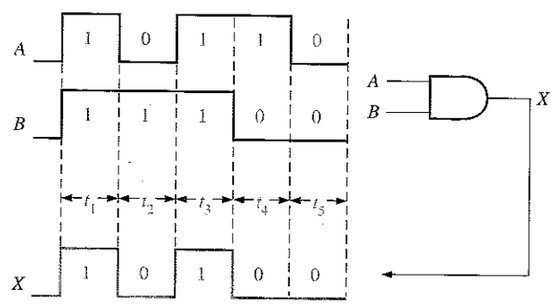


FIGURA 3.10
Ejemplo de funcionamiento de una puerta AND con trenes de impulsos, y cronograma que muestra las relaciones entre las entradas y la salida.

EJEMPLO 3.3 Si se aplican las formas de onda *A* y *B* de la Figura 3.11 a las entradas de una puerta AND, ¿cuál es la forma de onda resultante de salida?

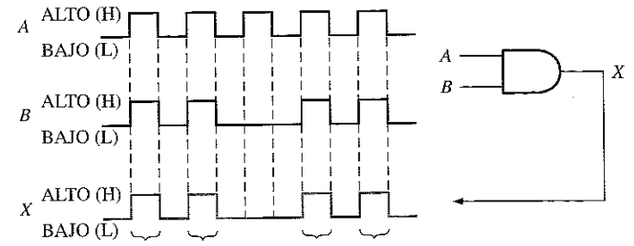


FIGURA 3.11
A y *B* están a nivel ALTO durante los cuatro intervalos de tiempo. Por tanto *X* está a nivel ALTO.

Solución. La forma de onda de salida *X* sólo está a nivel ALTO cuando *A* y *B* están a nivel ALTO, tal y como se muestra en el diagrama de tiempos de la Figura 3.11.

Problema relacionado. Determinar la forma de onda de salida y dibujar el diagrama de tiempos si los impulsos segundo y cuarto de la señal *A* de la Figura 3.11 se reemplazan ambos por un nivel BAJO.

Es importante recordar que cuando se analiza el funcionamiento con trenes de impulsos de las puertas lógicas, hay que poner especial cuidado en las relaciones de tiempo de todas las entradas entre sí y con la salida.

EJEMPLO 3.4 Para las dos formas de onda de entrada, *A* y *B*, de la Figura 3.12, dibujar la onda de salida mostrando su relación con las entradas.

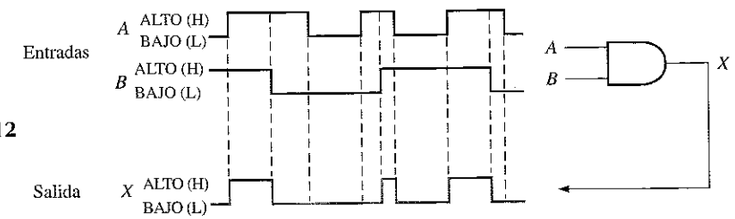


FIGURA 3.12

Solución. La onda de salida está a nivel ALTO sólo cuando ambas entradas están a nivel ALTO, como se muestra en el diagrama de tiempos.

Problema relacionado. Obtener la onda de salida si la entrada *B* de la puerta AND de la Figura 3.12 siempre está a nivel ALTO.

EJEMPLO 3.5 Para la puerta AND de 3 entradas de la Figura 3.13, determinar la forma de onda de salida con respecto a las entradas.

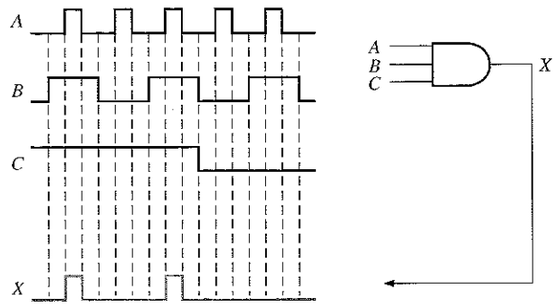


FIGURA 3.13

Solución. La onda de salida X de una puerta AND de 3 entradas está a nivel ALTO sólo cuando las tres entradas, A, B y C están a nivel ALTO.

Problema relacionado. ¿Cuál es la onda de salida de la puerta AND de la Figura 3.13 si la entrada C está siempre a nivel ALTO?

Expresiones lógicas para la puerta AND

La función lógica AND de dos variables se representa matemáticamente colocando un punto entre las dos variables, $A \cdot B$, o simplemente escribiendo las letras juntas sin el punto, AB. Normalmente, utilizaremos esta última notación, ya que es más cómoda de escribir.

La **multiplicación booleana** sigue las mismas reglas básicas que gobiernan la multiplicación binaria, que hemos tratado en el Capítulo 2 y son las siguientes:

$$0 \cdot 0 = 0$$
$$0 \cdot 1 = 0$$
$$1 \cdot 0 = 0$$
$$1 \cdot 1 = 1$$

La multiplicación booleana es lo mismo que la función AND.

El funcionamiento de una puerta AND de dos entradas puede expresarse en forma de ecuación como sigue: si una variable de entrada es A y la otra variable es B, y la variable de salida es X, entonces la expresión booleana es

$$X = AB$$

La Figura 3.14(a) muestra la puerta con las variables de entrada y de salida indicadas.

Para extender la expresión AND a más de dos variables de entrada, simplemente utilice una nueva letra para cada variable de entrada. Por ejemplo, la función de una puerta AND de 3 entradas se puede expresar así: $X = ABC$, donde A, B y C son las

variables de entrada. La expresión para una puerta AND de 4 entradas será $X = ABCD$, y así sucesivamente. Las partes (b) y (c) de la Figura 3.14 muestran, respectivamente, puertas AND de tres y cuatro variables de entrada.

NOTAS INFORMÁTICAS

Las computadoras pueden utilizar todas las operaciones lógicas básicas cuando tienen que manipular de forma selectiva ciertos bits pertenecientes a uno o más bytes de datos. La manipulación selectiva de bits se lleva a cabo mediante una *máscara*. Por ejemplo, para borrar (poner a 0) los cuatro bits de la derecha de un byte de datos manteniendo los cuatro bits de la izquierda, aplique la operación AND al byte de datos y al valor 11110000. Observe que cualquier bit que se opere (AND) con cero dará 0 y cualquier byte al que se aplique la operación AND con el valor 1 quedará como está. Si se aplica la operación AND al valor 10101010 con la máscara 11110000, el resultado es 10100000.

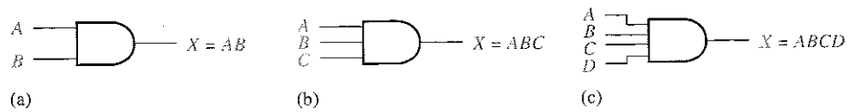


FIGURA 3.14
Expresión booleana para puertas AND con dos, tres y cuatro entradas.

Se puede evaluar el funcionamiento de una puerta AND utilizando las expresiones booleanas que facilitan la salida. Por ejemplo, cada variable de entrada puede ser 1 o 0, luego para evaluar la puerta AND de dos entradas, se sustituyen dichos valores en la ecuación de salida, $X = AB$, como se muestra en la Tabla 3.4. Esta evaluación muestra que la salida X de una puerta AND es 1 (nivel ALTO) sólo cuando ambas entradas son 1 (nivel ALTO). Se puede hacer un análisis similar para cualquier número de variables de entrada.

TABLA 3.4

A	B	$AB = X$
0	0	$0 \cdot 0 = 0$
0	1	$0 \cdot 1 = 0$
1	0	$1 \cdot 0 = 0$
1	1	$1 \cdot 1 = 1$

Ejemplos de aplicación

La puerta AND como un dispositivo de habilitación/inhibición. Una aplicación común de la puerta AND es **habilitar** (enable) o permitir el paso de una señal (tren de impulsos) de un punto a otro en determinados instantes, e inhibir o impedir el paso en otros instantes.

Un ejemplo sencillo de este particular uso de la puerta AND se muestra en la Figura 3.15, donde la puerta AND controla el paso de una señal (A) a un contador digital. El propósito de este circuito es medir la frecuencia de la señal A. El impulso de habilitación tiene un ancho de exactamente 1 s. Cuando la entrada de habilitación está a nivel ALTO, la señal A pasa a través de la puerta hasta el contador, y cuando la entrada de habilitación está a nivel BAJO, se impide el paso de la señal a través de la puerta.

Durante el intervalo de habilitación de 1 segundo, un cierto número de impulsos de la señal A pasan a través de la puerta AND hasta el contador. El número de impulsos que pasa durante 1 segundo es igual a la frecuencia de la señal. Por ejemplo, la Figura 3.15 muestra una señal en la que seis impulsos duran un segundo, lo que representa una frecuencia de 6 Hz. Si pasan 1.000 impulsos a través de la puerta en el intervalo de 1 segundo del impulso de habilitación, serán 1.000 impulsos/segundo, es decir una frecuencia de 1.000 Hz.

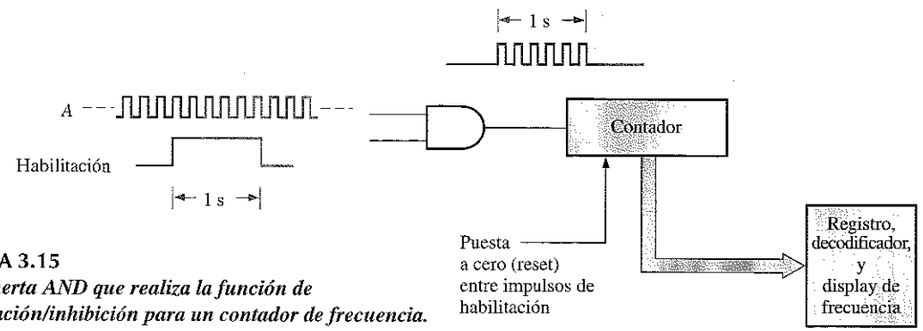


FIGURA 3.15
Una puerta AND que realiza la función de habilitación/inhibición para un contador de frecuencia.

El contador cuenta el número de impulsos por segundo y genera una salida binaria que pasa al circuito decodificador y al display, en el que se genera una lectura de la frecuencia. El impulso de habilitación se repite en determinados instantes, en los que se lleva a cabo un nuevo recuento por si la frecuencia ha variado, y el nuevo valor se presentará en el display. Entre los impulsos de habilitación, el contador se pone a cero para reinicializarse cada vez que se produce un impulso de habilitación. La frecuencia actual se almacena en un registro, de modo que el display no se vea afectado al poner a cero el contador.

Un sistema de alarma para el cinturón de seguridad. En la Figura 3.16, se usa una puerta AND en un sencillo sistema de alarma para el cinturón de seguridad del coche, el cual detecta cuándo el interruptor de arranque se ha activado y (AND) el cinturón de seguridad no está abrochado. Si el interruptor de arranque se ha activado, la entrada *A* de la puerta AND se pone a nivel ALTO. Si el cinturón de seguridad no está correctamente abrochado, la entrada *B* de la puerta AND se pone a nivel ALTO. También cuando el interruptor de arranque se activa, se inicializa un temporizador que pone a nivel ALTO la entrada *C* durante 30 segundos. Si estas tres condiciones se cumplen, es decir, si el interruptor de arranque está activado y (AND) el cinturón de seguridad está desabrochado y (AND) el temporizador está corriendo, la salida de la puerta AND se pone a nivel ALTO, y una alarma audible se activa para advertir al conductor.

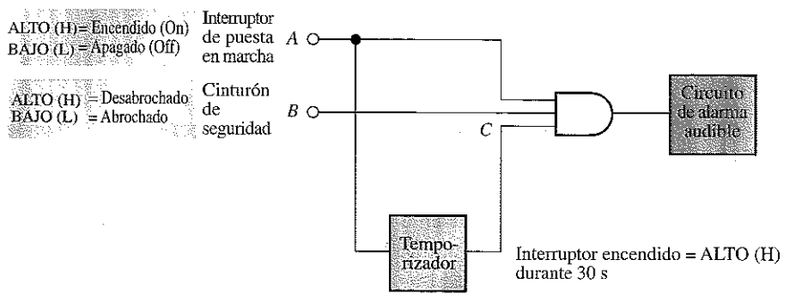


FIGURA 3.16
Un sencillo circuito de alarma para cinturón de seguridad utilizando una puerta AND.

REVISIÓN DE LA SECCIÓN 3.2

1. ¿Cuándo se pone a nivel ALTO la salida de una puerta AND?
2. ¿Cuándo se pone a nivel BAJO la salida de una puerta AND?
3. Describir la tabla de verdad para una puerta AND de cinco entradas.

3.3 ■ LA PUERTA OR

La puerta OR es otra de las puertas básicas con las que se construyen todas las funciones lógicas. Una puerta OR puede tener dos o más entradas y realiza la operación que se conoce como suma lógica.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Identificar una puerta OR mediante su símbolo distintivo y su símbolo rectangular. ■ Describir la operación lógica de una puerta OR. ■ Generar la tabla de verdad de una puerta OR con cualquier número de entradas. ■ Generar el diagrama de tiempos de una puerta OR para cualquier forma de onda especificada en sus entradas. ■ Escribir la expresión lógica de una puerta OR con cualquier número de entradas. ■ Desarrollar ejemplos de aplicaciones de la puerta OR.

Una **puerta OR** tiene dos o más entradas y una salida, como indican los símbolos lógicos estándar de la Figura 3.17, en la que se muestran puertas OR con dos entradas. Una puerta OR puede tener cualquier número de entradas mayor o igual que dos. Aunque se presentan ambos tipos de símbolos, distintivo y rectangular, en este texto se utilizará el símbolo distintivo de la puerta OR.

(a) Símbolo distintivo. (b) Símbolo rectangular con el identificador OR (≥ 1).

FIGURA 3.17
Símbolos lógicos estándar de la puerta OR con dos entradas (Estándar ANSI/IEEE 91-1984).

Operación lógica de la puerta OR

La **puerta OR** genera un nivel ALTO a la salida cuando *cualquiera* de sus entradas está a nivel ALTO. La salida se pone a nivel BAJO sólo cuando todas las entradas están a nivel BAJO. Por tanto, el propósito de una puerta OR es determinar cuándo una o más de sus entradas están a nivel ALTO y generar una salida a nivel ALTO que indique esta condición. Las entradas de la puerta OR de dos entradas de la Figura 3.17 están etiquetadas como *A* y *B*, y la salida como *X*. Podemos establecer el funcionamiento de la puerta como sigue:

En una puerta OR, la salida *X* es un nivel ALTO si cualquiera de las entradas, *A* o *B*, o ambas, están a nivel ALTO; *X* es un nivel BAJO si ambas entradas, *A* y *B*, están a nivel BAJO.

El nivel ALTO es el nivel de salida activo o verdadero para la puerta OR. La Figura 3.18 ilustra la operación lógica para una puerta OR de dos entradas, indicando las cuatro posibles combinaciones de entrada.

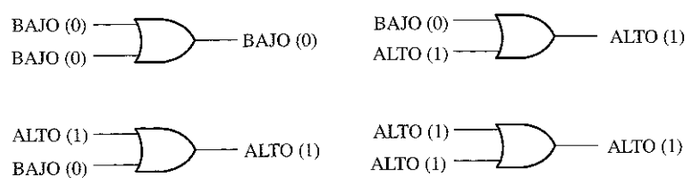


FIGURA 3.18
Todos los posibles niveles lógicos para una puerta OR de 2 entradas.

Tabla de verdad de una puerta OR

En la Tabla 3.5 se describe el funcionamiento lógico de una puerta OR de dos entradas. Esta tabla de verdad puede extenderse a cualquier número de entradas e, independientemente del número de entradas, la salida es un nivel ALTO cuando una o más entradas están a nivel ALTO.

TABLA 3.5
Tabla de verdad para una puerta OR de dos entradas.

Entradas		Salida
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	1

1 ≡ ALTO (H), 0 ≡ BAJO (L).

Funcionamiento con trenes de impulsos

Ahora vamos a ver el funcionamiento de una puerta OR con trenes de impulsos como entradas, teniendo en mente su modo de operación lógico. De nuevo, lo importante en el análisis del funcionamiento de la puerta con trenes de impulsos en las entradas es la relación de tiempos de todas las señales implicadas. Por ejemplo, en la Figura 3.19, las entradas *A* y *B* están a nivel ALTO (1) durante el intervalo t_1, haciendo que la salida esté a nivel ALTO (1). Durante el intervalo t_2, la entrada *A* está a nivel BAJO (0), pero, puesto que la entrada *B* está a nivel ALTO (1), la salida estará a nivel ALTO (1). Durante el intervalo t_3, ambas entradas están a nivel BAJO (0), luego en este intervalo la salida estará a nivel BAJO (0). Durante el intervalo t_4, la salida está a nivel ALTO (1) ya que la entrada *A* está a nivel ALTO (1).

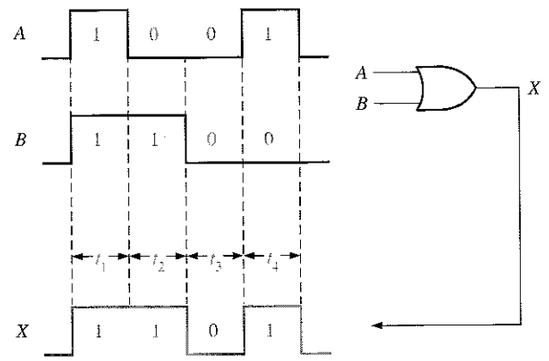
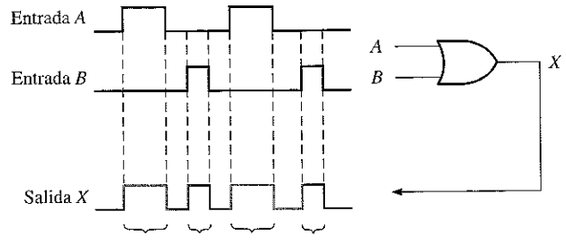


FIGURA 3.19
Ejemplo de funcionamiento de la puerta OR con trenes de impulsos junto con el cronograma que muestra la relación entre las entradas y la salida.

En este ejemplo, simplemente hemos aplicado la tabla de verdad de la puerta OR para cada uno de los intervalos de tiempo durante los cuales los niveles no cambiaban. Los Ejemplos del 3.6 al 3.8 ilustran el funcionamiento de la puerta OR con diferentes señales de entrada.

EJEMPLO 3.6 Si se aplican las dos señales de entrada, *A* y *B*, de la Figura 3.20 a la puerta OR, ¿cuál es la señal de salida resultante?



Cuando cualquiera de las entradas o ambas están a nivel ALTO, la salida se pone a nivel ALTO.

FIGURA 3.20

Solución. La señal de salida *X* de una puerta OR de dos entradas será un nivel ALTO cuando una o ambas entradas estén a nivel ALTO, tal como muestra el diagrama de tiempos. En este caso, ambas entradas nunca están al tiempo a nivel ALTO.

Problema relacionado. Determinar la señal de salida y dibujar el diagrama de tiempos si la entrada *A* se cambia de modo que está a nivel ALTO desde el inicio del primer impulso hasta el final del segundo impulso.

EJEMPLO 3.7 Para las dos ondas de entrada, A y B, de la Figura 3.21, dibujar la onda de salida indicando su relación respecto a las entradas.

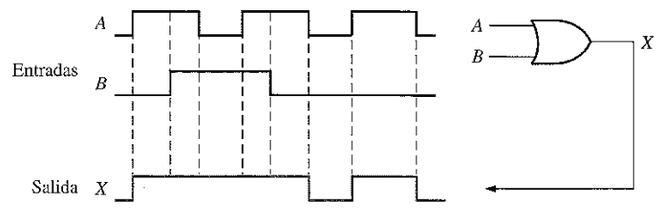


FIGURA 3.21

Solución. Cuando una o ambas entradas están a nivel ALTO, la salida estará a nivel ALTO, como muestra la señal de salida X en el diagrama de tiempos.

Problema relacionado. Determinar la señal de salida y dibujar el cronograma si el impulso central de la entrada A se sustituye por un nivel BAJO.

EJEMPLO 3.8 Para la puerta OR de 3 entradas de la Figura 3.22, determinar la señal de salida respecto de las entradas en función del tiempo.

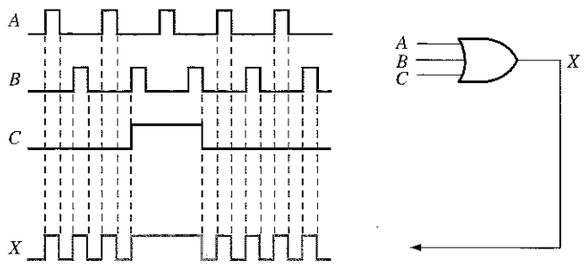


FIGURA 3.22

Solución. La salida está a nivel ALTO cuando una o más entradas están a nivel ALTO, como muestra la señal de salida X en el diagrama de tiempos.

Problema relacionado. Determinar la señal de salida y dibujar el diagrama de tiempos si la entrada C está siempre a nivel BAJO.

Expresiones lógicas de la puerta OR

La función lógica OR de dos variables se representa matemáticamente mediante un signo + entre las dos variables, por ejemplo, $A + B$.

La suma en el álgebra de Boole implica variables cuyos valores son, bien el binario 1 o el binario 0. Las reglas básicas de la **suma booleana** son las siguientes:

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 1$$

La suma booleana es lo mismo que la función OR.

Obsérvese que la suma booleana difiere de la suma binaria en el caso en que se suman dos 1s. En la suma booleana no existe acarreo.

El funcionamiento de una puerta OR de 2 entradas se puede expresar como sigue: si una variable de entrada es A y la otra variable de entrada es B, y la variable de salida es X, entonces la expresión booleana es

$$X = A + B$$

La Figura 3.23(a) muestra el símbolo lógico de la puerta indicando las variables de entradas y de salida.

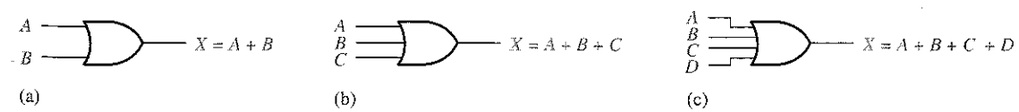


FIGURA 3.23
Expresiones booleanas de las puertas OR con dos, tres y cuatro entradas.

Para ampliar la expresión OR a más de dos variables de entrada, se usa una nueva letra para cada variable adicional. Por ejemplo, la función de una puerta OR de 3 entradas se puede expresar como $X = A + B + C$, y la expresión para una puerta OR de 4 entradas sería $X = A + B + C + D$, y así sucesivamente. Las ilustraciones (b) y (c) de la Figura 3.23 muestran, respectivamente, las variables de entrada de las puertas OR de tres y cuatro entradas.

Como se muestra en la Tabla 3.6 para una puerta OR de dos entradas, el funcionamiento de la puerta OR se puede evaluar utilizando las expresiones booleanas para la salida X, sustituyendo todas las posibles combinaciones de 1 y 0 en las variables de entrada. Esta evaluación muestra que la salida X de una puerta OR es un 1 (nivel ALTO) cuando una o más de sus entradas son 1 (nivel ALTO). Un análisis similar se puede hacer para las puertas OR con cualquier número de variables de entrada.

NOTAS INFORMÁTICAS Otra operación de enmascaramiento que se usa en la programación de computadoras para hacer que determinados bits de un byte de datos sean igual a 1 (lo que se denomina activación) de forma selectiva, no afectando a otros bits, se lleva a cabo con la operación OR. Se utiliza una máscara que contenga un 1 en cualquier posición en la que un bit de datos desee ponerse a 1. Por ejemplo, si desea forzar que el bit más significativo de un byte de datos sea igual a 1, dejando los demás bits como están, puede aplicar la operación OR al byte de datos con la máscara 10000000.

TABLA 3.6

A	*B*	*A* + *B* = *X*
0	0	0 + 0 = 0
0	1	0 + 1 = 1
1	0	1 + 0 = 1
1	1	1 + 1 = 1

Ejemplo de aplicación

En la Figura 3.24 se presenta parte de un sistema de alarma y detección de intrusión simplificado. Este sistema se podría utilizar en una habitación de una casa: una habitación con dos ventanas y una puerta. Los sensores son interruptores magnéticos que producen un nivel de salida ALTO cuando se abre la puerta (o las ventanas) y una salida a nivel BAJO cuando se cierra. Mientras que las ventanas y la puerta están aseguradas, los interruptores están cerrados y las tres entradas de la puerta OR están a nivel BAJO. Cuando se abre una de las ventanas o la puerta, en la entrada correspondiente de la puerta OR se genera un nivel ALTO y la salida de la puerta lógica se pone a nivel ALTO. Entonces se activa el circuito de alarma para advertir de la intrusión.

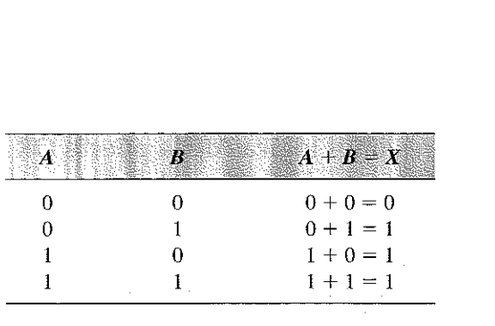


FIGURA 3.24
Un sistema de detección de intrusión simplificado que utiliza una puerta OR.

REVISIÓN DE LA SECCIÓN 3.3

1. ¿Cuándo se pone a nivel ALTO la salida de una puerta OR?
2. ¿Cuándo se pone a nivel BAJO la salida de una puerta OR?
3. Escribir la tabla de verdad para una puerta OR de tres entradas.

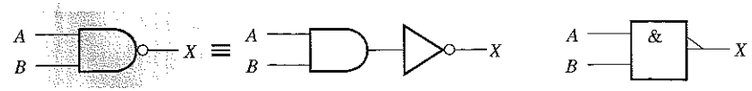
3.4 ■ LA PUERTA NAND

La puerta NAND es un elemento lógico popular, debido a que se puede utilizar como una puerta universal, es decir, las puertas NAND se pueden combinar para implementar las operaciones de las puertas AND, OR y del inversor. En el Capítulo 5, se examinará la propiedad universal de la puerta NAND.

Al finalizar esta sección, el lector deberá ser capaz de:

- Identificar una puerta NAND mediante su símbolo distintivo y su símbolo rectangular.
- Describir la operación lógica de una puerta NAND. ■ Desarrollar la tabla de verdad de una puerta NAND con cualquier número de entradas. ■ Generar el cronograma de una puerta NAND para cualquier forma de onda especificada en sus entradas.
- Escribir la expresión lógica de una puerta NAND con cualquier número de entradas.
- Describir la operación de la puerta NAND en términos de la puerta equivalente negativa-OR. ■ Desarrollar ejemplos de aplicaciones de la puerta NAND.

El término *NAND* es una contracción de NOT-AND, e implica una función AND con la salida complementada (negada). En la Figura 3.25(a) se muestra el símbolo lógico estándar para la puerta NAND de 2 entradas y su equivalente empleando los símbolos de la puerta AND seguida de un inversor. En la parte (b) se muestra el símbolo rectangular.



(a) Símbolo distintivo, puerta NAND de dos entradas y su equivalente NOT/AND.

(b) Símbolo rectangular, puerta NAND de dos entradas con indicador de polaridad.

FIGURA 3.25
Símbolos lógicos estándar de la puerta NAND (ANSI-/IEEE 91-1984).

Operación lógica de la puerta NAND

La **puerta NAND** genera una salida a nivel BAJO sólo cuando todas las entradas están a nivel ALTO. Cuando cualquiera de las entradas está a nivel BAJO, la salida se pondrá a nivel ALTO. Para el caso concreto de la puerta NAND de dos entradas, como muestra la Figura 3.25, con la designación *A* y *B* para las entradas y *X* para la salida, el modo de operación se puede establecer como sigue:

> **En una puerta NAND de dos entradas, la salida *X* es un nivel BAJO si las entradas *A* y *B* están a nivel ALTO; *X* es un nivel ALTO si *A* o *B* están a nivel BAJO o si ambas, *A* y *B*, están a nivel BAJO.**

Observe que esta operación, en términos de nivel de salida, es la opuesta a la operación lógica AND. En una puerta NAND, el nivel BAJO (0) es el nivel activo o verdadero de salida, como indica el círculo de la salida. La Figura 3.26 ilustra la operación

lógica de una puerta NAND de dos entradas, para las cuatro posibles combinaciones, y la Tabla 3.7 es la tabla de verdad, que resume la operación lógica de la puerta NAND de dos entradas.

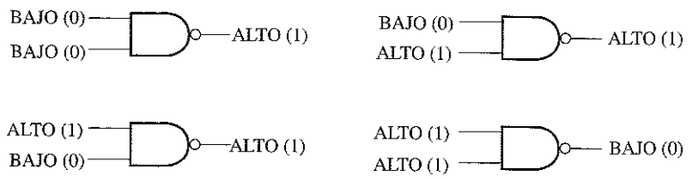


FIGURA 3.26
Operación lógica de la puerta NAND de 2 entradas.

TABLA 3.7
Tabla de verdad de la puerta NAND de 2 entradas.

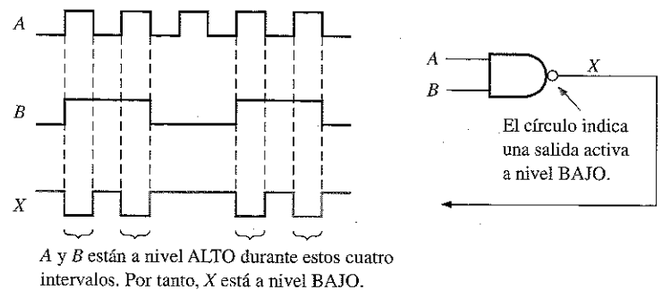
Entradas		Salida
A	***B***	***X***
0	0	1
0	1	1
1	0	1
1	1	0

1 = ALTO (H), 0 = BAJO (L).

Funcionamiento con trenes de impulsos

Veamos ahora el funcionamiento de la puerta NAND con un tren de impulsos. Recuérdese de su tabla de verdad que la salida se pone a nivel BAJO sólo cuando todas las entradas están a nivel ALTO. Los Ejemplos 3.9 y 3.10 ilustran el funcionamiento con trenes de impulsos.

EJEMPLO 3.9 Si a las entradas de una puerta NAND se aplican las formas de onda *A* y *B* de la Figura 3.27, determinar la forma de onda resultante de salida.



A y *B* están a nivel ALTO durante estos cuatro intervalos. Por tanto, *X* está a nivel BAJO.

FIGURA 3.27

Solución. La señal de salida *X* está a nivel BAJO sólo durante cuatro intervalos de tiempo, cuando ambas entradas, *A* y *B*, están a nivel ALTO, como muestra el diagrama de tiempos.

Problema relacionado. Determinar la onda de salida y dibujar el diagrama de tiempos si se invierte la entrada *B*.

EJEMPLO 3.10 Obtener la forma de onda de salida para la puerta NAND de tres entradas de la Figura 3.28, donde además se presenta el diagrama de tiempos de las entradas.

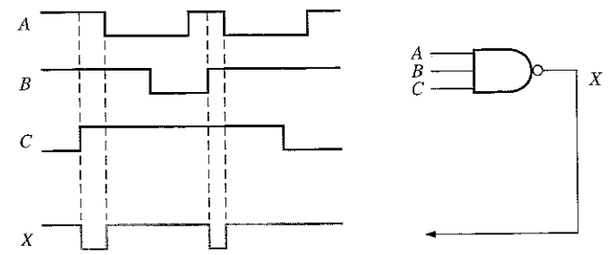


FIGURA 3.28

Solución. La señal de salida *X* está a nivel BAJO sólo cuando las tres entradas están a nivel ALTO, tal como se muestra en el diagrama de tiempos.

Problema relacionado. Determinar la forma de onda de salida y dibujar el diagrama de tiempos si se invierte la señal de entrada *A*.

Operación equivalente negativa-OR de la puerta NAND. Es inherente al funcionamiento de la puerta NAND el hecho de que una o más entradas a nivel BAJO dan lugar a una salida a nivel ALTO. La Tabla 3.7 indica que la salida *X* está a nivel ALTO (1) cuando cualquiera de las entradas, *A* y *B*, están a nivel BAJO (0). Desde este punto de vista, la puerta NAND se puede usar para realizar la operación OR que requiere una o más entradas a nivel BAJO, para generar una salida a nivel ALTO. Este modo de operación se denomina **negativa-OR**. En este contexto, el término *negativa* significa que las entradas se definen para que su estado activo o verdadero sea un nivel BAJO.

En una puerta NAND de dos entradas que funciona como una puerta negativa-OR, la salida *X* es un nivel ALTO si cualquiera de las entradas, *A* o *B*, es un nivel BAJO, o si ambas entradas, *A* y *B*, están a nivel BAJO.

Cuando la puerta NAND se emplea con uno o más niveles bajos en sus entradas en lugar de con niveles altos, se trata como una puerta negativa-OR y se representa con

el símbolo lógico estándar de la Figura 3.29. Aunque los dos símbolos de esta figura representan la misma puerta física, sirven para definir el papel o el modo de operación en una aplicación particular, como se ilustra en los Ejemplos 3.11 hasta 3.13.

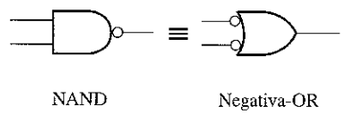


FIGURA 3.29
Símbolos estándar para representar las dos operaciones equivalentes de la puerta NAND.

EJEMPLO 3.11 Una planta de fabricación utiliza dos tanques para almacenar un determinado líquido químico que se requiere en un proceso de fabricación. Cada tanque dispone de un sensor que detecta cuándo el nivel del líquido cae al 25% del total. Los sensores generan una tensión de 5 V cuando los tanques están llenos por encima del 25%. Cuando el volumen de líquido en el tanque cae por debajo del 25%, el sensor genera un nivel de 0 V.

En el panel indicador se requiere un diodo emisor de luz (LED, light-emitting diode) verde que indique que el nivel de ambos tanques está por encima del 25%. Como se indica, se puede utilizar una puerta NAND para implementar esta función.

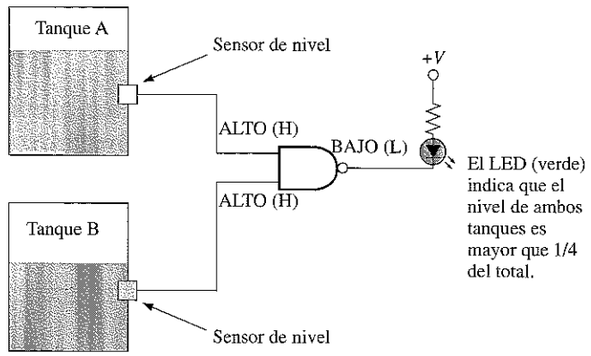


FIGURA 3.30

Solución. La Figura 3.30 muestra una puerta NAND de dos entradas conectada a los sensores de nivel del tanque y la salida conectada al panel indicador. La operación puede definirse como sigue: si el nivel del tanque *A* y el nivel del tanque *B* están por encima de un cuarto del total, el LED se enciende.

Mientras que la salida de *ambos* sensores esté a nivel ALTO (5 V), quiere decir que el nivel de los tanques está por encima del 25% del volumen total, y la salida de la puerta NAND está a nivel BAJO (0 V). El circuito del LED verde se activa para una tensión a nivel BAJO.

Problema relacionado. ¿Cómo se puede modificar el circuito de la Figura 3.30 para controlar el nivel de tres tanques en lugar de dos?

EJEMPLO 3.12 El supervisor del proceso de fabricación descrito en el Ejemplo 3.11 ha decidido que sería preferible disponer de un LED rojo encendido cuando al menos el nivel de líquido de uno de los tanques estuviera por debajo del 25%, y que el LED verde se encendiera cuando el nivel en ambos tanques estuviera por encima de dicho límite. Veamos cómo se puede implementar este requisito.

Solución. La Figura 3.31 muestra que la puerta NAND está funcionando como una puerta negativa-OR, para detectar la ocurrencia de que al menos una de las entradas esté a nivel BAJO. Un sensor genera un nivel BAJO si el volumen en su tanque es del 25% o menor. Cuando esto ocurre, la salida de la puerta pasa a nivel ALTO, por lo que el LED rojo del panel indicador se enciende para un nivel de tensión ALTO. La operación puede definirse del siguiente modo: si el tanque *A* o el tanque *B*, *o* ambos están por debajo del 25% del total, entonces el LED se enciende.

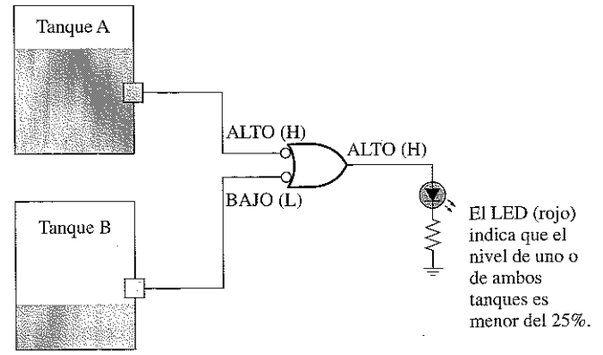


FIGURA 3.31

Obsérvese que en este ejemplo y en el Ejemplo 3.11, se usa la misma puerta NAND de dos entradas, pero en el esquema se ha empleado un símbolo de puerta diferente para ilustrar la diferencia funcional de las puertas NAND y negativa-OR.

Problema relacionado. ¿Cómo se puede modificar el circuito de la Figura 3.31 para controlar cuatro tanques en lugar de dos?

EJEMPLO 3.13 Para la puerta NAND de cuatro entradas de la Figura 3.32, que opera como una puerta negativa-OR, determinar la salida con respecto a sus entradas.

Solución. La forma de onda de salida *X* está a nivel ALTO siempre que una entrada esté a nivel BAJO, como se indica en el diagrama de tiempos.

Problema relacionado. Determinar la forma de onda de salida, si se invierte la señal de entrada *A* antes de aplicarla a la puerta.

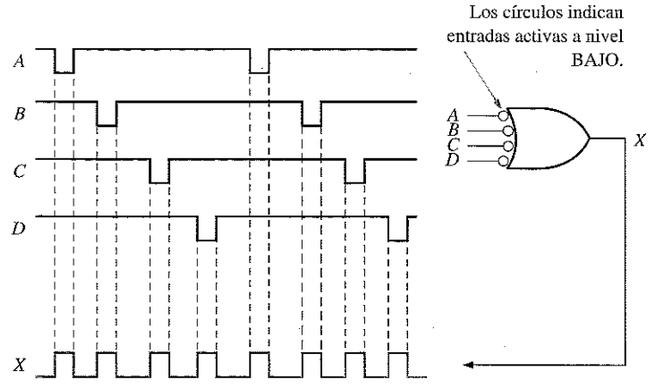


FIGURA 3.32

Expresiones lógicas para la puerta NAND

La expresión booleana para la puerta NAND de dos entradas es:

$$X = \overline{AB}$$

Esta expresión significa que las dos variables de entrada, A y B, se multiplican (AND) primero y luego se complementan, tal y como indica la barra sobre la expresión lógica correspondiente a AND. Ésta es una descripción lógica en forma de ecuación de la operación de una puerta NAND con dos entradas. Si se evalúa esta expresión para todos los posibles valores de las dos variables de entrada, el resultado que se obtiene es el indicado en la Tabla 3.8.

TABLA 3.8

A	B	$\overline{AB} = X$
0	0	$\overline{0 \cdot 0} = \overline{0} = 1$
0	1	$\overline{0 \cdot 1} = \overline{0} = 1$
1	0	$\overline{1 \cdot 0} = \overline{0} = 1$
1	1	$\overline{1 \cdot 1} = \overline{1} = 0$

Una vez determinada la expresión lógica para una función lógica dada, esta función se puede evaluar para todos los valores posibles de las variables. La evaluación facilita de forma exacta la salida del circuito lógico para cada una de las condiciones de entrada y, por tanto, le proporciona una descripción completa de la operación lógica del circuito. La expresión para NAND se puede extender a más de dos variables de entrada, incluyendo letras adicionales para representar las otras variables.

REVISIÓN DE LA SECCIÓN 3.4

1. ¿Cuándo está a nivel BAJO la salida de una puerta NAND?
2. ¿Cuándo está a nivel ALTO la salida de una puerta NAND?
3. Describir las diferencias funcionales entre una puerta NAND y una puerta negativa-OR. ¿Tienen ambas la misma tabla de verdad?
4. Escribir la expresión de salida para una puerta NAND con tres entradas, A, B y C.

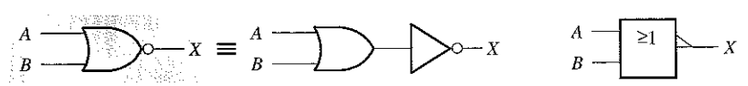
3.5 ■ LA PUERTA NOR

La puerta NOR, al igual que la puerta NAND, es un útil elemento lógico porque también se puede emplear como una puerta universal, es decir, las puertas NOR se pueden usar en combinación para implementar las operaciones AND, OR y del inversor. En el Capítulo 5 se examinará la propiedad universal de la puerta NOR.

Al finalizar esta sección, el lector deberá ser capaz de:

- Identificar una puerta NOR mediante su símbolo distintivo y su símbolo rectangular.
- Describir la operación lógica de una puerta NOR. ■ Desarrollar la tabla de verdad de una puerta NOR con cualquier número de entradas. ■ Generar el diagrama de tiempos de una puerta NOR para cualquier forma de onda especificada en sus entradas.
- Escribir la expresión lógica de una puerta NOR con cualquier número de entradas.
- Describir la operación de la puerta NOR en términos de su equivalente negativa-AND.
- Desarrollar ejemplos de aplicaciones de la puerta NOR.

El término *NOR* es una contracción de NOT-OR e implica una función OR con la salida invertida (complementada). En la Figura 3.33(a) se muestra el símbolo lógico estándar para la puerta NOR de 2 entradas y su equivalente empleando los símbolos de la puerta OR seguida de un inversor. En la parte (b) se muestra el símbolo rectangular.



(a) Símbolo distintivo, puerta NOR de 2 entradas y sus equivalentes NOT/OR

(b) Símbolo rectangular, puerta NOR de 2 entradas con indicador de polaridad

FIGURA 3.33
Símbolo lógico estándar para la puerta NOR (ANSI/IEEE Std. 91-1984).

Operación lógica de la puerta NOR

La **puerta NOR** genera una salida a nivel BAJO cuando *cualquiera* de sus entradas está a nivel ALTO. Cuando todas sus entradas estén a nivel BAJO, la salida se pondrá a nivel ALTO. Para el caso concreto de la puerta NOR de dos entradas, que se muestra en la Figura 3.33, con la designación A y B para las entradas y X para la salida, el modo de operación se puede establecer como sigue:

En una puerta NOR de dos entradas: la salida *X* es un nivel BAJO si cualquiera de sus entradas *A* o *B* está a nivel ALTO, o si ambas entradas *A* y *B* están a nivel ALTO; *X* es un nivel ALTO si *A* y *B* están a nivel BAJO.

Esta operación genera un nivel de salida opuesto al que genera la puerta OR. En una puerta NOR, el nivel BAJO es el nivel activo o verdadero de salida, como indica el círculo de la salida. La Figura 3.34 ilustra la operación lógica de una puerta NOR de dos entradas, para las cuatro posibles combinaciones de entrada, y la Tabla 3.9 es la tabla de verdad para la puerta NOR de dos entradas.

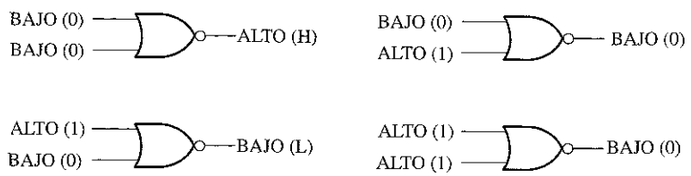


FIGURA 3.34
Operación lógica de la puerta NOR de 2 entradas.

TABLA 3.9
Tabla de verdad de la puerta NOR de 2 entradas.

Entradas		Salida
A	***B***	***X***
0	0	1
0	1	0
1	0	0
1	1	0

1 = ALTO (H), 0 = BAJO (L).

Funcionamiento con trenes de impulsos

Los dos ejemplos siguientes ilustran la operación lógica de la puerta NOR con entradas que son trenes de impulsos. De nuevo, como en los demás tipos de puertas, simplemente deberemos aplicar la tabla de verdad para determinar la forma de onda de salida de acuerdo con el diagrama de tiempos de las entradas.

EJEMPLO 3.14 Si se aplican a la puerta NOR las dos señales de la Figura 3.35, ¿cómo es la señal de salida que se obtiene?

Solución. Cuando cualquier entrada de la puerta NOR está a nivel ALTO, la salida es un nivel BAJO, como se ve en la forma de onda de salida *X* del diagrama de tiempos.

Problema relacionado. Invertir la entrada *B* y determinar la forma de onda de salida para esas entradas.

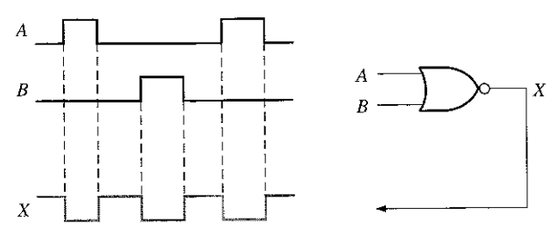


FIGURA 3.35

EJEMPLO 3.15 Determinar la señal de salida para la puerta NOR de la Figura 3.36, teniendo en cuenta el diagrama de tiempos correspondiente a las entradas.

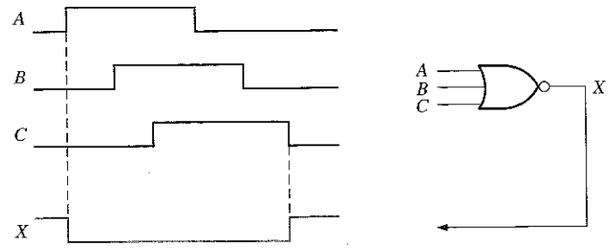


FIGURA 3.36

Solución. La salida *X* se pone a nivel BAJO cuando cualquier entrada está a nivel ALTO, como muestra la señal de salida *X* en el diagrama de tiempos.

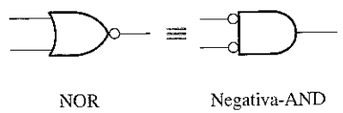
Problema relacionado. Con las entradas *B* y *C* invertidas, determinar la salida y dibujar el cronograma.

Operación equivalente negativa-AND de la puerta NOR. La puerta NOR, al igual que la puerta NAND, posee otro aspecto de su funcionamiento que es inherente a su función lógica. La Tabla 3.9 muestra que se genera un nivel ALTO en la salida sólo si todas las entradas están a nivel BAJO. Desde este punto de vista, la puerta NOR se puede utilizar como una operación AND cuyas entradas están a nivel BAJO y generan una salida a nivel ALTO. Este modo de operación es denominada **negativa-AND**. En este contexto, el término *negativa* significa que las entradas están definidas para que su estado activo o verdadero sea un nivel BAJO.

En una puerta NOR de dos entradas que funciona como una puerta negativa-AND, la salida *X* es un nivel ALTO cuando ambas entradas, *A* y *B*, están a nivel BAJO.

Cuando se quiere que la puerta NOR presente todas sus entradas a nivel BAJO en lugar de presentar uno o más niveles altos, se aplica la puerta negativa-AND y se representa mediante el símbolo estándar de la Figura 3.37. Es importante recordar que los dos símbolos de la Figura 3.37 representan la misma puerta física, y sólo sirven para distinguir entre los dos modos de operación lógicos. Los tres ejemplos siguientes ilustran esto.

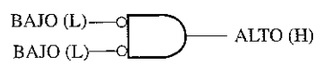
FIGURA 3.37
Símbolos estándar que representan dos operaciones equivalentes de la puerta NOR.



EJEMPLO 3.16 Se necesita un dispositivo para indicar cuándo se producen simultáneamente dos niveles de entrada bajos que dan lugar a un nivel de salida ALTO. Especificar el dispositivo.

Solución. Para generar una salida a nivel ALTO cuando ambas entradas están a nivel BAJO se requiere una puerta negativa-AND, como se muestra en la Figura 3.38.

FIGURA 3.38



Problema relacionado. Se necesita un dispositivo para indicar una salida a nivel BAJO, cuando uno o dos niveles altos se aplican a sus entradas. Especificar el dispositivo.

EJEMPLO 3.17 Como parte del sistema de monitorización funcional de un avión, se requiere un circuito para indicar el estado del tren de aterrizaje antes de tomar tierra. Se enciende un LED verde si los tres mecanismos de aterrizaje están correctamente extendidos cuando el interruptor para "bajar el tren de aterrizaje" se ha activado. Un LED rojo se enciende si cualquiera de los mecanismos falla al extenderse antes de aterrizar. Cuando uno de los mecanismos se extiende, el sensor correspondiente genera una tensión a nivel BAJO. Cuando uno de los mecanismos del tren de aterrizaje se retrae, su sensor genera una tensión a nivel ALTO. Implementar un circuito que cumpla estos requisitos.

Solución. La alimentación se aplica al circuito sólo cuando el interruptor para "bajar el tren de aterrizaje" se activa. Se debe utilizar una puerta NOR para cada uno de los dos requisitos, tal y como se muestra en la Figura 3.39. Una puerta NOR funciona como una puerta negativa-AND para detectar un nivel BAJO procedente de cada uno de los sensores de los mecanismos de aterrizaje. Cuando las tres entradas de la puerta están a nivel BAJO, los tres mecanismos (ruedas) están correctamente extendidos y dan lugar a una salida a nivel ALTO en la puerta negativa-AND, lo que hace que se encienda el LED verde. La otra puerta NOR funciona como tal para detectar si una o más de las ruedas de aterrizaje permanecen retraídas; la salida a nivel ALTO procedente del sensor se detecta mediante la puerta NOR, que da lugar a una salida a nivel BAJO que enciende el LED rojo de aviso.

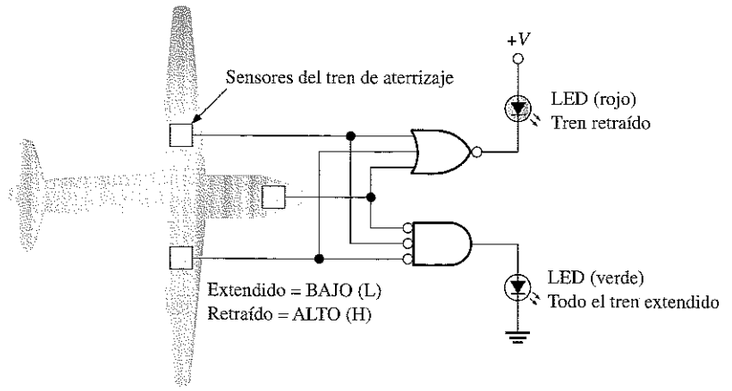


FIGURA 3.39

Problema relacionado. ¿Qué tipo de puerta se debería usar para detectar si las tres ruedas del tren de aterrizaje están escondidas después de despegar? Suponer que se precisa un nivel de salida BAJO para activar un LED.

Consejos Prácticos

Cuando se excita una carga tal como un LED con una puerta lógica, debería consultarse la hoja de características del fabricante para conocer los parámetros máximos de excitación (corriente de salida). Un CI de puerta lógica normal puede no ser capaz de manipular la corriente necesaria para determinadas cargas, como pueden ser algunos LED. Hay disponibles puertas lógicas con salida tipo buffer, tal como una salida en colector abierto o en drenador abierto en muchos tipos de configuraciones de circuitos integrados. La capacidad de corriente de salida de las puertas lógicas en CI típicas están limitadas en el rango de los μA o de los mA. Por ejemplo, las puertas estándar TTL pueden manipular corrientes de salida de hasta 16 mA. La mayor parte de los LED precisan corrientes en el rango comprendido entre los 10 mA y 50 mA.

EJEMPLO 3.18 Para una puerta NOR de cuatro entradas que funciona como una puerta negativa-AND como la de la Figura 3.40, determinar la salida en función de las entradas.

Solución. En cualquier instante en el que todas las entradas estén a nivel BAJO, la salida estará a nivel ALTO, siendo la forma de onda de salida *X* la indicada en el cronograma.

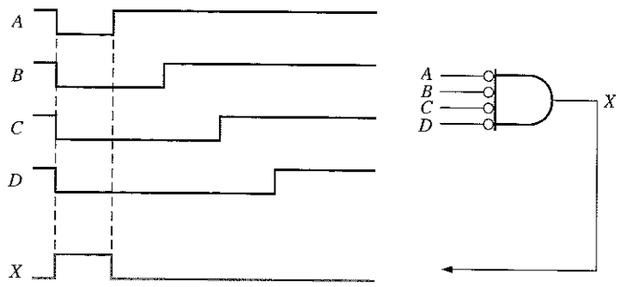


FIGURA 3.40

Problema relacionado. Determinar la salida para la entrada *D* invertida y dibujar el diagrama de tiempos.

Expresiones lógicas para la puerta NOR

La expresión booleana para la salida de una puerta NOR de dos entradas se expresa así:

$$X = \overline{A + B}$$

Esta ecuación indica que las dos variables de entrada primero se suman (operación OR) y luego se complementan, tal y como indica la barra sobre la expresión lógica OR. Evaluando esta expresión, se obtienen los resultados mostrados en la Tabla 3.10. La expresión NOR puede extenderse a más de dos variables de entrada, incluyendo letras adicionales para representar otras variables.

TABLA 3.10

A	*B*	$\overline{A+B} = X$
0	0	$\overline{0+0} = \overline{0} = 1$
0	1	$\overline{0+1} = \overline{1} = 0$
1	0	$\overline{1+0} = \overline{1} = 0$
1	1	$\overline{1+1} = \overline{1} = 0$

REVISIÓN DE LA SECCIÓN 3.5

1. ¿Cuándo está a nivel ALTO la salida de una puerta NOR?
2. ¿Cuándo está a nivel BAJO la salida de una puerta NOR?
3. Describir la diferencia funcional entre una puerta NOR y una puerta negativa-AND. ¿Tienen ambas la misma tabla de verdad?
4. Escribir las expresiones de salida para una puerta NOR de tres entradas, siendo las variables de entrada, *A*, *B* y *C*.

3.6 ■ PUERTAS OR-EXCLUSIVA Y NOR-EXCLUSIVA

Las puertas OR-exclusiva y NOR-exclusiva se forman mediante la combinación de otras puertas que ya hemos tratado, como se verá en el Capítulo 5. Sin embargo, debido a su importancia fundamental en muchas aplicaciones, estas puertas se tratan como elementos lógicos básicos con su propio símbolo único.

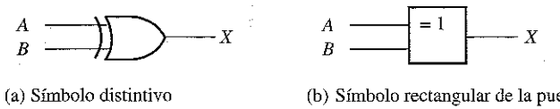
Al finalizar esta sección, el lector deberá ser capaz de:

■ Identificar las puertas OR-exclusiva y NOR-exclusiva mediante su símbolo distintivo y su símbolo rectangular. ■ Describir la operación lógica de las puertas OR-exclusiva y NOR-exclusiva. ■ Desarrollar las tablas de verdad de las puertas OR-exclusiva y NOR-exclusiva. ■ Generar el diagrama de tiempos de las puertas OR-exclusiva y NOR-exclusiva para cualquier forma de onda especificada en sus entradas. ■ Desarrollar ejemplos de aplicaciones de las puertas NOR-exclusiva y OR-exclusiva.

La puerta OR-exclusiva

En la Figura 3.41 se muestran los símbolos estándar para la puerta OR-exclusiva (XOR). La puerta XOR tiene sólo dos entradas.

NOTAS INFORMÁTICAS Varias puertas OR-exclusivas conectadas para formar un circuito sumador permiten que una computadora realice las operaciones de suma, sustracción, multiplicación y división en su unidad aritmético lógica (ALU). Una puerta OR-exclusiva se implementa con los circuitos lógicos básicos de las puertas AND, OR y NOT.



(a) Símbolo distintivo (b) Símbolo rectangular de la puerta XOR con el símbolo que la identifica (=1)

FIGURA 3.41
Símbolos lógicos estándar de la puerta OR-exclusiva.

La salida de una **puerta OR-exclusiva** se pone a nivel ALTO *sólo* cuando las dos entradas están a niveles lógicos opuestos. Esta operación se puede expresar, en función de dos entradas *A* y *B* y una salida *X*, del siguiente modo:

En una puerta OR-exclusiva, la salida *X* es un nivel ALTO si la entrada *A* está a nivel BAJO y la entrada *B* está a nivel ALTO; o si la entrada *A* está a nivel ALTO y la entrada *B* está a nivel BAJO; *X* es un nivel BAJO si tanto *A* como *B* están a nivel ALTO o BAJO.

En la Figura 3.42 se ilustran las cuatro posibles combinaciones de entrada y las salidas resultantes para la puerta XOR. El nivel ALTO es el nivel activo o verdadero de salida y sólo se produce cuando las entradas están a niveles opuestos. La operación lógica de la puerta XOR se resume en la tabla de verdad mostrada en la Tabla 3.11.

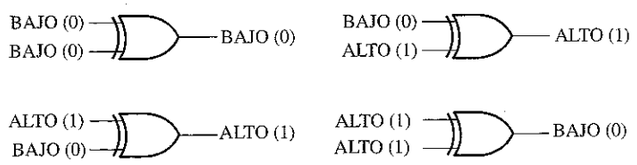


FIGURA 3.42
Todos los niveles lógicos posibles para una puerta OR-exclusiva.

TABLA 3.11
Tabla de verdad de la puerta OR-exclusiva.

Entradas		Salida
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	0

EJEMPLO 3.19 Un cierto sistema está formado por dos circuitos idénticos que funcionan en paralelo. Mientras que ambos funcionan correctamente, las salidas de los dos circuitos son iguales. Si uno de los circuitos falla, las salidas serán niveles opuestos en ese instante. Establecer un método para detectar que se ha producido un fallo en uno de los circuitos.

Solución. Las salidas de los circuitos se conectan a las entradas de una puerta XOR, tal y como muestra la Figura 3.43. Un fallo en cualquiera de los circuitos hace que las entradas de la puerta XOR tengan niveles opuestos. Esta condición da lugar a nivel ALTO a la salida de la puerta XOR, que indica que uno de los circuitos ha fallado.

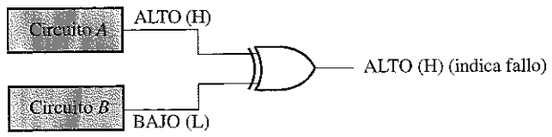


FIGURA 3.43

Problema relacionado. ¿Detectará siempre la puerta OR-exclusiva fallos simultáneos en los dos circuitos de la Figura 3.43? Si la respuesta es no, ¿bajo qué condiciones los detectará?

La puerta NOR-exclusiva

En la Figura 3.44 se presentan los símbolos estándar de la **puerta NOR-exclusiva** (XNOR). Al igual que la puerta XOR, la puerta XNOR sólo tiene dos entradas. El círculo en la salida del símbolo de la puerta XNOR indica que su salida es la opuesta a la de la puerta XOR. Cuando dos niveles lógicos de entrada son opuestos, la salida de la puerta NOR-exclusiva es un nivel BAJO. La operación se puede expresar del siguiente modo (*A* y *B* son las entradas, y *X* es la salida).

En una puerta NOR-exclusiva, la salida *X* es un nivel BAJO si la entrada *A* está a nivel BAJO y la entrada *B* está a nivel ALTO, o si *A* está a nivel ALTO y *B* está a nivel BAJO; *X* es un nivel ALTO si *A* y *B* están ambas a nivel ALTO o BAJO.

FIGURA 3.44
Símbolos lógicos estándar para la puerta NOR-exclusiva.

En la Figura 3.45 se muestran las cuatro posibles combinaciones de entrada y las salidas resultantes para la puerta XNOR. La operación lógica se resume en la Tabla 3.12. Obsérvese que la salida es un nivel ALTO cuando las dos entradas están al mismo nivel.

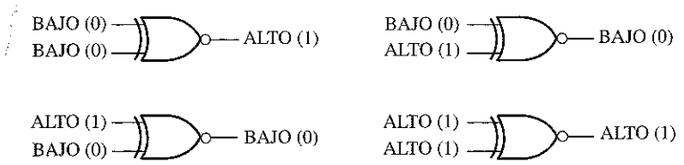


FIGURA 3.45
Todos los niveles lógicos posibles para una puerta NOR-exclusiva.

TABLA 3.12
Tabla de verdad de la puerta NOR-exclusiva.

Entradas		Salida
A	*B*	*X*
0	0	1
0	1	0
1	0	0
1	1	1

Funcionamiento con trenes de impulsos

Al igual que hemos hecho con las puertas anteriores, vamos a examinar el funcionamiento de las puertas XOR y XNOR para trenes de impulsos en sus entradas. Como antes, aplicamos la tabla de verdad para cada intervalo de tiempo de los trenes de impulsos, como se muestra en la Figura 3.46 para una puerta XOR. Puede comprobarse que las señales de entrada *A* y *B* tienen niveles opuestos en los intervalos t_2 y t_4.

Por tanto, la salida X se pone a nivel ALTO durante estos dos intervalos. En los intervalos t_1 y t_3, ambas entradas están al mismo nivel, BAJO o ALTO, y la salida se pone a nivel BAJO para estos impulsos, como muestra el diagrama de tiempos.

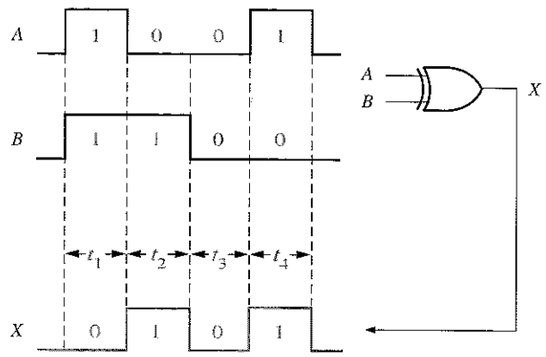


FIGURA 3.46
Ejemplo de funcionamiento de la puerta OR-exclusiva con trenes de impulsos.

EJEMPLO 3.20 Determinar las formas de onda de salida para la puerta XOR y la puerta XNOR, dadas las formas de onda de entrada A y B de la Figura 3.47.

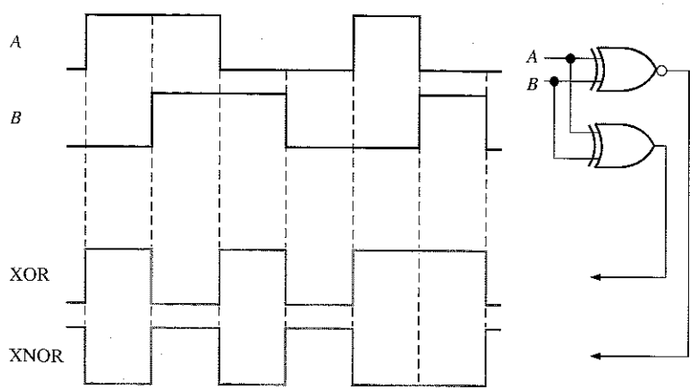


FIGURA 3.47

Solución. Las formas de onda de salida son las que se indican en la Figura 3.47. Obsérvese que la salida XOR está a nivel ALTO sólo cuando las entradas están a niveles opuestos. La salida XNOR está a nivel ALTO sólo cuando ambas entradas son iguales.

Problema relacionado. Determinar la forma de onda de salida si las dos entradas, A y B, se invierten.

Ejemplo de aplicación

La puerta OR-exclusiva se puede utilizar como sumador de dos bits. Recuérdese del Capítulo 2 que las reglas básicas de la suma binaria son: 0 + 0 = 0, 0 + 1 = 1, 1 + 0 = 1 y 1 +1 = 10. Un examen de la tabla de verdad de la puerta XOR demostrará que su salida es la suma binaria de los dos bits de entrada. En el caso en que ambas entradas sean 1, la salida de la suma es 0 y se pierde el acarreo de 1. En el Capítulo 6, se verá cómo se combinan puertas XOR para realizar circuitos sumadores completos. La Figura 3.48 presenta la puerta XOR utilizada como sumador básico.

Bits de entrada		Salida (suma)	
A	B	Σ	
0	0	0	
0	1	1	
1	0	1	
1	1	0	(sin acarreo de 1)

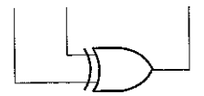


FIGURA 3.48
Puerta XOR utilizada como sumador de dos bits.

REVISIÓN DE LA SECCIÓN 3.6

1. ¿Cuándo está a nivel ALTO la salida de una puerta XOR?
2. ¿Cuándo está a nivel ALTO la salida de una puerta XNOR?
3. ¿Cómo se puede utilizar una puerta XOR para detectar que dos bits son diferentes?

3.7 ■ PUERTAS LÓGICAS INTEGRADAS

Existen varias tecnologías de circuitos integrados digitales que se usan para implementar las puertas lógicas básicas. Dos de estas tecnologías, CMOS y TTL, son las más extendidas y otra de ellas, la ECL, se usa para aplicaciones más especializadas. Las operaciones lógicas NOT, AND, OR, NAND, NOR y OR-exclusiva son las mismas, independientemente de la tecnología de circuitos integrados que se utilice; es decir, una puerta AND tiene la misma función lógica se implemente con la tecnología CMOS, TTL o ECL. En esta sección no vamos a tratar las tecnologías de circuitos integrados en el nivel de componentes de circuito. No obstante, aquellos lectores que deseen estudiar este tema con mayor profundidad pueden consultar el Capítulo 15: "Tecnologías de circuitos integrados", para ver información detallada de los circuitos.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Identificar las series más comunes CMOS y TTL. ■ Comparar las tecnologías CMOS y TTL en términos de tipos de dispositivos y parámetros de funcionamiento. ■ Definir el *retardo de propagación*. ■ Definir la *disipación de potencia*. ■ Definir

el *fan-out*. ■ Definir el *producto velocidad-potencia*. ■ Interpretar la información básica de una hoja de características.

El término **CMOS** corresponde a *Complementary Metal-Oxide Semiconductor* (metal-óxido semiconductor complementario) y se implementa con un tipo de transistor de efecto de campo. **TTL** corresponde a la transistor-transistor lógic y se implementa mediante transistores bipolares. **ECL** corresponde a *Emitter-Coupled Logic* (lógica de emisor acoplado) y también se implementa mediante la tecnología **bipolar**. En esta sección, vamos a tratar únicamente las tecnologías CMOS y TTL, ya que son las tecnologías de CI dominantes actualmente. Tenga en cuenta que CMOS y TTL sólo difieren en el tipo de componentes de circuito y los valores de los parámetros, y no en las operaciones lógicas básicas. Una puerta AND CMOS realiza la misma operación lógica que una puerta AND TTL. Esto también es cierto para todas las operaciones lógicas básicas restantes. La diferencia entre CMOS y TTL se encuentra en las características de funcionamiento, tal como la velocidad de conmutación (retardo de propagación), la disipación de potencia, la inmunidad al ruido y otros parámetros.

CMOS

Existe poco desacuerdo sobre cuál es la tecnología de circuitos, CMOS o TTL, más ampliamente utilizada. Parece que CMOS está comenzando a ser la tecnología dominante y podría reemplazar a la tecnología TTL. Esto es especialmente cierto en el caso de los circuitos de gran escala de integración y en los microprocesadores, todos los cuales se implementan con tecnología MOS. Aunque TTL ha dominado durante muchos años, principalmente debido a sus altas velocidades de conmutación y a una enorme variedad de tipos de dispositivos, la tecnología CMOS siempre ha tenido la ventaja de ofrecer una mucho menor disipación de potencia. Las velocidades de conmutación de CMOS han mejorado extremadamente y ahora pueden competir con TTL, a la vez que la baja disipación de potencia y otros factores deseables se han mantenido a medida què la tecnología avanzaba.

Series CMOS. Las dos categorías de CMOS en términos de tensión de alimentación continua son la serie CMOS de 5 V y la serie CMOS de 3,3 V. Esta última es el desarrollo más reciente y es el resultado de un esfuerzo por reducir la disipación de potencia. Puesto que la disipación de potencia es proporcional al cuadrado de la tensión, una reducción de 5 V a 3,3 V disminuye la potencia en un 34%, sin que el resto de los factores varíen.

Dentro de cada categoría según la tensión de alimentación, hay disponibles varias series de puertas lógicas CMOS. Estas series pertenecientes a la familia CMOS difieren en sus características de funcionamiento y se designan mediante los prefijos 74 o 54, seguidos de una letra o letras que indican la serie y, a continuación, un número que indica el tipo de dispositivo lógico. El prefijo 74 indica que se trata de un dispositivo comercial de propósito general, y el prefijo 54 indica que es un dispositivo militar para aplicaciones en entornos más exigentes. En este libro sólo haremos referencia a los dispositivos que llevan el prefijo 74. La serie básica CMOS de 5 V y sus denominaciones son las siguientes:

- 74HC y 74HCT. CMOS de alta velocidad (la "T" indica compatibilidad TTL)
- 74AC y 74ACT. CMOS avanzada
- 74AHC y 74AHCT. CMOS de alta velocidad avanzada

La serie básica CMOS de 3,3 V y sus denominaciones son las siguientes:

- 74LV. CMOS de baja tensión
- 74LVC. CMOS de baja tensión.
- 74ALVC. CMOS de baja tensión avanzada.

Además de la serie 74, todavía existe la serie 4000, que es una tecnología CMOS más antigua y de baja velocidad, aunque su uso está limitado. Además de las series CMOS "puras" existen series que combinan ambas tecnologías, CMOS y TTL, que se denominan BiCMOS. La serie básica BiCMOS y sus denominaciones son las siguientes:

- 74BCT. BiCMOS
- 74ABT. BiCMOS avanzada.
- 74LVT. BiCMOS de baja tensión.
- 74ALB. BiCMOS de baja tensión.

TTL

La tecnología TTL ha sido y es todavía una tecnología de circuitos integrados digitales muy popular. Una ventaja de esta tecnología es que no es sensible a las descargas electrostáticas como lo es la tecnología CMOS y, por tanto, es más práctica en la realización de experimentos de laboratorio y la elaboración de prototipos, ya que no es necesario preocuparse por los problemas de manipulación.

Series TTL. Al igual que con la tecnología CMOS, hay disponibles varias series de puertas lógicas TTL, las cuales operan todas ellas con 5 V de alimentación de continua. Estas series pertenecientes a la familia TTL difieren en sus características de funcionamiento y se denominan mediante los prefijos 74 o 54 seguidos por una letra o letras que indican la serie y un número que indica el tipo de dispositivo lógico de la serie. Un circuito integrado TTL puede distinguirse de un circuito integrado CMOS por las letras que siguen a los prefijos 74 y 54.
Las series básicas TTL y sus denominaciones son las siguientes:

- 74: TTL estándar (sin letra).
- 74S: TTL Schottky.
- 74AS: TTL Schottky avanzada.
- 74LS: TTL Schottky de baja potencia.
- 74ALS: TTL Schottky de baja potencia avanzada.
- 74F: TTL rápida.

Tipos de puertas lógicas integradas

Todas las operaciones lógicas básicas: NOT, AND, OR, NAND, NOR y OR-exclusiva (XOR) están disponibles en las tecnologías de circuitos integrados. También están disponibles puertas con salida buffer para excitar cargas que requieran altas corrientes. Los tipos de configuraciones de puerta normalmente disponibles en los circuitos

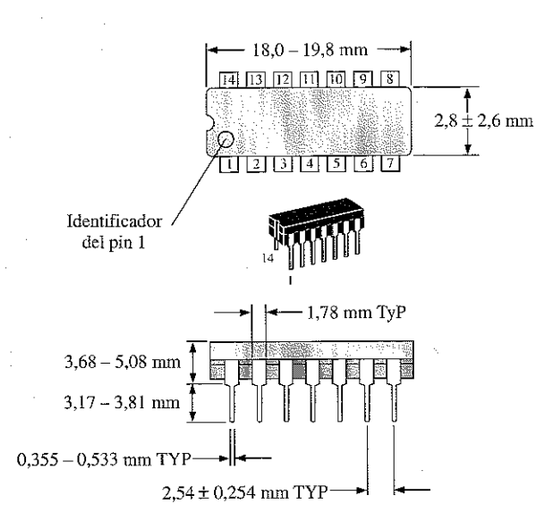
integrados se identifican mediante los dos o tres dígitos finales de la designación de la serie. Por ejemplo, 74LS04 es un circuito integrado inversor séxtuple **Schottky** de baja potencia. Algunas de las configuraciones de puertas lógicas habituales y sus dígitos de identificación estándar son los siguientes:

- Cuádruple NAND de dos entradas: **00**
- Cuádruple NOR de dos entradas: **02**
- Inversor séxtuple: **04**
- Cuádruple AND de dos entradas: **08**
- Triple NAND de tres entradas: **10**
- Triple AND de tres entradas: **11**
- Doble NAND de cuatro entradas: **20**
- Doble AND de dos entradas: **21**
- Triple NOR de tres entradas: **27**
- NAND de ocho entradas: **30**
- Cuádruple OR de dos entradas: **32**
- Cuádruple XOR de dos entradas: **86**
- NAND única de trece entradas: **133**

Encapsulados de circuitos integrados. Todos los CMOS de la serie 74 son compatibles en su patillaje con respecto a los mismos tipos de dispositivos en tecnología TTL. Esto quiere decir que un circuito integrado digital CMOS, como el 74AHC00 (cuádruple NAND de 2 entradas), que contiene cuatro puertas NAND de 2 entradas en un único circuito integrado, tiene exactamente los mismos números de pin para cada entrada y salida que el correspondiente dispositivo TTL. En la Figura 3.49 se muestra los encapsulados típicos de circuitos integrados, el encapsulado DIP (dual in-line package) para montaje de inserción y el encapsulado SOIC (small-outline integrated circuit) para montaje superficial. En algunos casos, hay disponibles algunos otros tipos de encapsulados. Aunque no se muestra a escala, el encapsulado SOIC es significativamente más pequeño que el encapsulado DIP. Los diagramas de configuración de los pines para la mayor parte de los dispositivos lógicos enumerados anteriormente se muestran en la Figura 3.50.

Lógica monopuerta. Hay disponible una selección limitada de puertas CMOS en encapsulados monopuerta. Con una única puerta por chip, esta serie se vende en pequeños encapsulados de cinco pines que sirven para llevar a cabo las modificaciones de último momento para incluir determinados circuitos lógicos en zonas de alta densidad, cuando el espacio disponible es limitado.

Símbolos lógicos. Los símbolos lógicos para los circuitos integrados digitales utilizan los símbolos de puerta estándares y presentan el número de puertas en el encapsulado del circuito integrado y los números de pines asociados a cada puerta, así como los números de pines para V_{CC} y masa. En la Figura 3.51 se muestra un ejemplo para el inversor séxtuple y para la puerta cuádruple NAND de 2 entradas. Se muestran el diagrama lógico y el símbolo rectangular. Independientemente de la familia lógica, todos los dispositivos con el mismo sufijo tienen pines compatibles, es decir, tienen la misma disposición de números de pines. Por ejemplo, los dispositivos 7400, 74S00, 74LS00, 74ALS00, 74F00, 74HC00 y 74AHC00 son todos ellos circuitos integrados cuádruples NAND de 2 entradas compatibles en cuanto a sus pines.



(a) Encapsulado DIP de 14 pines para montaje de inserción.

(b) Encapsulado SOIC de 14 pines para montaje superficial.

FIGURA 3.49
Encapsulados típicos DIP y SOIC con sus dimensiones básicas y la numeración de los pines.

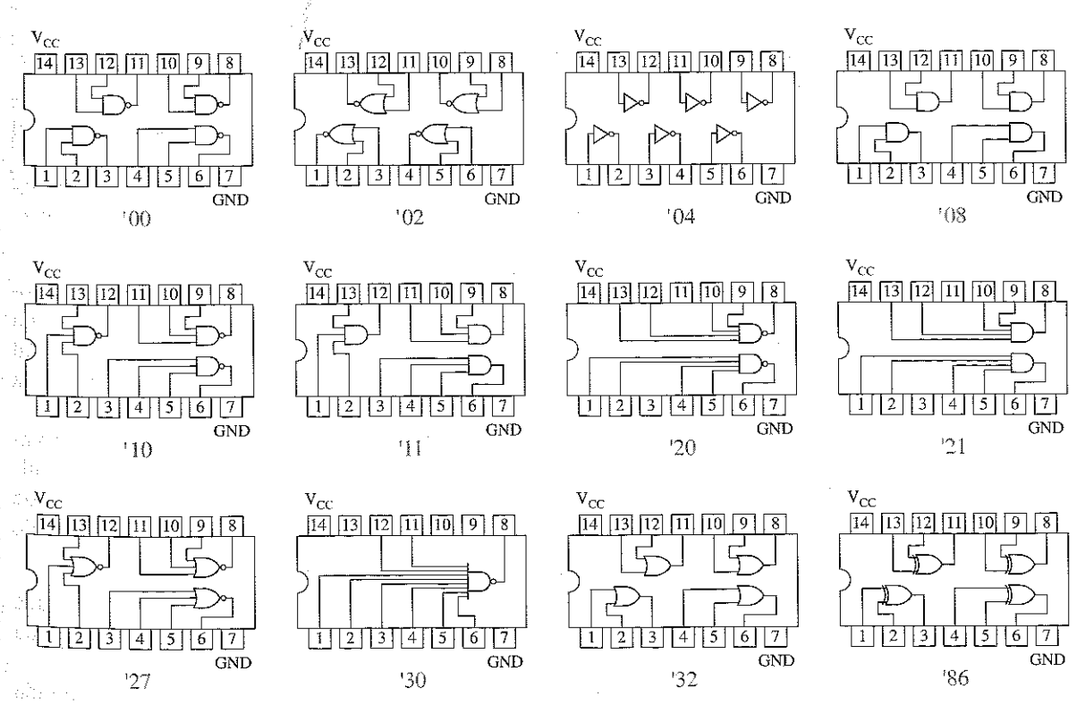


FIGURA 3.50
Diagramas de configuración de los pines para algunas de las configuraciones de puertas integradas más comunes.

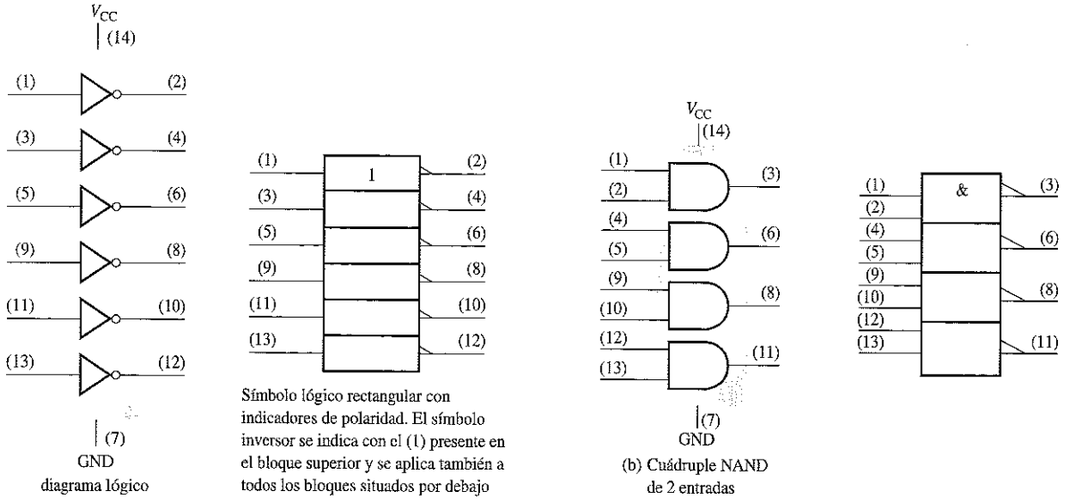


FIGURA 3.51
Símbolos lógicos para el inversor séxtuple (sufijo 04) y la puerta NAND cuádruple de 2 entradas (sufijo 00). El símbolo se aplica al mismo dispositivo en cualquier serie CMOS o TTL.

Características y parámetros de funcionamiento

Existen varios puntos que definen el funcionamiento de un circuito lógico. Las características de funcionamiento son la velocidad de conmutación medida en términos del retardo de propagación, la disipación de potencia, el fan-out o capacidad de excitación, el producto velocidad-potencia, la tensión de alimentación continua y los niveles lógicos de entrada/salida. Vamos a definir cada una de estas características y después compararemos el funcionamiento de CMOS y TTL.

Tiempo de retardo de propagación. Este parámetro limita la frecuencia o velocidad de conmutación a la que un circuito lógico puede operar. Cuando se aplican a los circuitos lógicos, los términos *baja velocidad* y *alta velocidad* hacen referencia al retardo de propagación. Cuanto menor sea el tiempo de propagación, mayor será la velocidad del circuito y mayor será la frecuencia a la que puede operar.

El **tiempo de retardo de propagación**, t_P, de una puerta lógica es el intervalo de tiempo entre la aplicación de un impulso de entrada y la aparición del impulso de salida resultante. Existen dos medidas diferentes del tiempo de retardo de propagación asociado con una puerta lógica, que se aplican a todos los tipos de puertas básicas:

- t_{PHL}: es el tiempo entre un punto de referencia especificado en el impulso de entrada y el correspondiente punto de referencia en el impulso de salida, cuando la salida cambia del nivel ALTO (H) al nivel BAJO (L).
- t_{PLH}: es el tiempo entre un punto de referencia especificado en el impulso de entrada y el correspondiente punto de referencia en el impulso de salida, cuando la salida cambia del nivel BAJO (L) al nivel ALTO (H).

EJEMPLO 3.21 Determinar los tiempos de propagación en el inversor de la Figura 3.52(a).

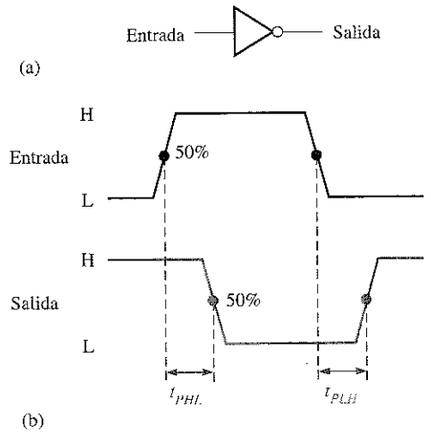


FIGURA 3.52

Solución. Los tiempos de propagación, t_{PHL} y t_{PLH}, se indican en la parte (b) de la figura. En este caso, los retrasos se miden entre los puntos de pendiente del 50% en los correspondientes flancos de los impulsos de entrada y salida. Los valores de t_{PHL} y t_{PLH} no necesariamente son iguales, pero, en la mayoría de los casos, sí lo son.

Problema relacionado. Una puerta lógica tiene una valor máximo especificado de 10 ns para t_{PHL} y t_{PLH}. Para otros tipos de puertas este valor es de 4 ns. ¿Cuál es la puerta que puede trabajar a la frecuencia mayor?

En las puertas TTL de la serie estándar, el retardo de propagación típico es de 11 ns y para las puertas de la serie F es de 3,3 ns. Para las puertas CMOS de la serie HCT, el retardo de propagación es 7 ns, para la serie AC es de 5 ns y para la serie ALVC es de 3 ns. Todos los valores especificados dependen de determinadas condiciones de operación, tal y como se establece en las hojas de características. Por propósitos de comparación, en un dispositivo ECL, el retardo de propagación típico es de 0,22 ns.

Tensión de alimentación continua (V_{CC}). La tensión de alimentación continua típica para CMOS puede ser 5 V o 3,3 V, dependiendo de la categoría. Una ventaja de CMOS es que las tensiones de alimentación pueden variar en un rango más amplio que los dispositivos TTL. Los CMOS de 5 V pueden tolerar variaciones de alimentación desde los 2 V a 6 V y aun así funcionarán adecuadamente, aunque el retardo de propagación y la disipación de potencia se vean significativamente afectadas. Los dispositivos CMOS de 3,3 V pueden operar con tensiones de alimentación desde 2 V hasta 3,6 V. La tensión de alimentación continua típica para dispositivos TTL es 5,0 V con un mínimo de 4,5 V y un máximo de 5,5 V.

Disipación de potencia. La **disipación de potencia**, P_D, de una puerta lógica es el producto de la tensión de alimentación continua y de la corriente media de alimentación. Normalmente, la corriente de alimentación cuando la salida de la puerta está a nivel BAJO es mayor que cuando la salida de la puerta está a nivel ALTO. Generalmente, las hojas de características del fabricante especifican la corriente de alimentación para el estado de salida BAJO como I_{CCL} y para el estado ALTO como I_{CCH}. La corriente media de alimentación se determina en función de un ciclo de trabajo del 50% (nivel de salida BAJO la mitad del tiempo y la otra mitad nivel de salida ALTO), por tanto la disipación de potencia media de una puerta lógica es

$$P_D = V_{CC}\left(\frac{I_{CCH} + I_{CCL}}{2}\right) \tag{3.2}$$

Las puertas de la serie CMOS tienen disipaciones de potencia muy bajas en comparación con las series TTL. Sin embargo, la disipación de potencia en los dispositivos CMOS depende de la frecuencia de funcionamiento. Para una frecuencia cero, la potencia está normalmente en el rango de los microwatios por puerta, y en la frecuencia máxima de funcionamiento puede estar en el rango de los miliwatios; por tanto, algunas veces la potencia se especifica para una frecuencia determinada. Por ejemplo, la serie HC tiene una potencia de 2,75 μW/puerta para una frecuencia igual a 0 Hz y de 600 μW/puerta para 1 MHz.

La disipación de potencia para los dispositivos TTL es independiente de la frecuencia. Por ejemplo, la serie ALS disipa 1,4 mW/puerta, independientemente de la frecuencia y la serie F disipa 6 mW/puerta.

Niveles lógicos de entrada y salida. V_{IL} es la tensión del nivel de entrada BAJO para una puerta lógica y V_{IH} es la tensión de entrada del nivel ALTO. Los dispositivos CMOS de 5 V aceptan una tensión máxima de 1,5 V para V_{IL} y una tensión mínima de 3,5 V para V_{IH}. Los dispositivos TTL aceptan una tensión máxima de 0,8 V para V_{IL} y una tensión mínima de 2 V para V_{IH}.

V_{OL} es la tensión de salida para el nivel BAJO y V_{OH} es la tensión de salida para el nivel ALTO. Para los dispositivos CMOS de 5 V, el valor máximo de V_{OL} es de 0,33 V y el valor mínimo para V_{OH} es de 4,4 V. Para los dispositivos TTL, el valor máximo V_{OL} es de 0,4 V y el mínimo V_{OH} es de 2,4 V. Todos los valores dependen de las condiciones de operación, tal y como se especifica en la hoja de características.

Producto velocidad-potencia (SPP). El parámetro **SPP (speed-power product)** puede utilizarse como una medida del funcionamiento de un circuito lógico que tiene en cuenta el retardo de propagación y la disipación de potencia. Es especialmente útil para comparar las distintas series de puertas lógicas de las familias CMOS o TTL o para comparar una puerta CMOS con una puerta TTL.

El producto SPP de un circuito lógico es igual al producto del retardo de propagación por la disipación de potencia, y se expresa en julios (J), que es una unidad de energía. La fórmula es

$$SPP = t_p P_D \tag{3.3}$$

EJEMPLO 3.22 Una determinada puerta tiene un retardo de propagación de 5 ns, I_{CCH} = 1 mA e I_{CCL} = 2,5 mA, con una tensión de alimentación continua de 5 V. Determinar el producto velocidad-potencia.

Solución

$$P_D = V_{CC}\left(\frac{I_{CCH} + I_{CCL}}{2}\right) = 5\ \text{V}\left(\frac{1\ \text{mA} + 2{,}5\ \text{mA}}{2}\right) = 5\ \text{V}(1{,}75\ \text{mA}) = 8{,}75\ \text{mW}$$

SPP = (5 ns)(8,75 mW) = **43,75 pJ**

Problema relacionado. Si el retardo de propagación de una puerta es 15 ns y su SPP es igual a 150 pJ, ¿cuál es su disipación de potencia media?

Fan-out y carga. El **fan-out** de una puerta lógica es el número máximo de entradas de la familia de circuitos integrados de la misma serie que la puerta puede excitar, manteniendo a la vez los niveles de salida dentro de los límites especificados. El fan-out es un parámetro importante sólo en la tecnología TTL. Dado que con los circuitos CMOS se asocian impedancias muy altas, el fan-out es muy alto, aunque depende de la frecuencia debido a los efectos capacitivos.

El fan-out se especifica en términos de **cargas unidad**. Una carga unidad para una puerta lógica es igual a una entrada de un circuito similar. Por ejemplo, una carga unidad para una puerta NAND 7400 es igual a *una* entrada a una puerta lógica en la serie estándar 74 (no necesariamente una puerta NAND). Puesto que, como se especifica en la hoja de características, la corriente para una entrada a nivel ALTO (I_{IH}) de una puerta 7400 es 40 μA y la corriente de una entrada a nivel BAJO (I_{IL}) es 1,6 mA, el número de cargas unidad que una puerta 7400 puede excitar en ambos estados de salida es

$$\text{Cargas unidad} = \frac{I_{OH}}{I_{IH}} = \frac{I_{OL}}{I_{IL}} = \frac{400\ \mu\text{A}}{40\ \mu\text{A}} = \frac{16\ \text{mA}}{1{,}6\ \text{mA}} = 10$$

La Figura 3.53 muestra puertas lógicas LS que excitan una serie de puertas con la misma tecnología de circuitos, donde el número de puertas depende de la tecnología de circuitos particular. Por ejemplo, como hemos visto, el número máximo de entradas de puerta (cargas unidad) que una puerta estándar TTL de la serie 74 puede excitar es 10. La mayor parte de las restantes series TTL, tal como la LS, pueden excitar 20 cargas unidad.

Comparación del retardo de propagación y la potencia en CMOS y TTL

La gráfica de la Figura 3.54 proporciona una comparación relativa de los retardos de propagación y las disipaciones de potencia para varias series CMOS de puertas lógicas. Cuanto más cortas son las barras, menor es el retardo de propagación (muy alta velocidad) y menor es la disipación de potencia. Por ejemplo, puede verse que la serie

ALVC es tres veces más rápida que la serie LV, que es la serie CMOS más lenta mostrada aquí. La gráfica también muestra que los CMOS de la serie AC son los que menos potencia disipan y que la serie AHC es la que más potencia disipa.

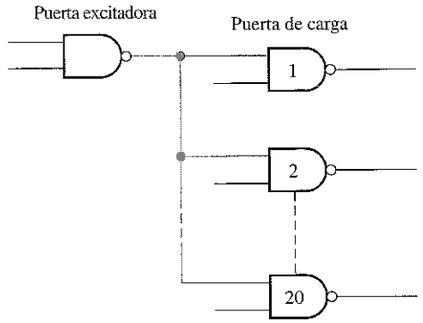


FIGURA 3.53
La salida de la puerta NAND LS TTL admite como carga máxima la entrada de 20 puertas LS TTL.

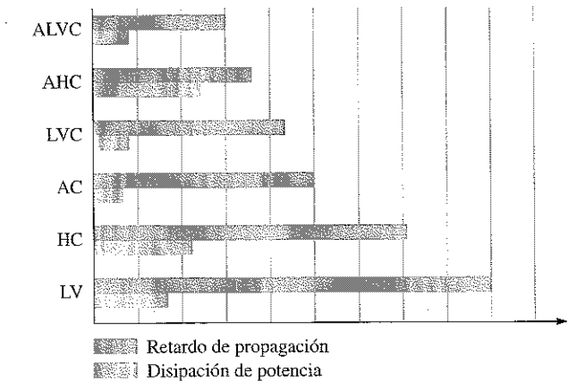


FIGURA 3.54
Comparación relativa de las velocidades de conmutación y disipaciones de potencia en la familia CMOS. Las barras gris oscuro indican el retardo de propagación y las barras gris claro indican la disipación de potencia.

La gráfica de la Figura 3.55 proporciona comparaciones relativas de los tiempos de propagación y disipaciones de potencia para diversas series TTL de puertas lógicas. La gráfica de la Figura 3.56 proporciona una comparación relativa de las velocidades de conmutación (retardos de propagación) para CMOS y TTL. Como puede verse, CMOS ofrece dispositivos que son más rápidos que el más rápido de los TTL. No se muestra la comparación de potencias porque los valores CMOS están en el rango de los μW y los valores TTL están en el rango de los mW.

FIGURA 3.55
Comparación relativa de las velocidades de conmutación y disipaciones de potencia de la familia TTL.

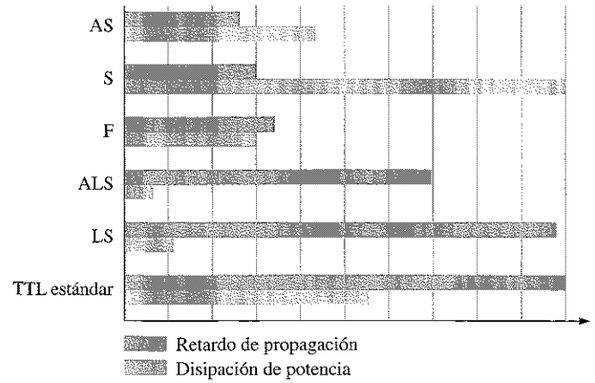
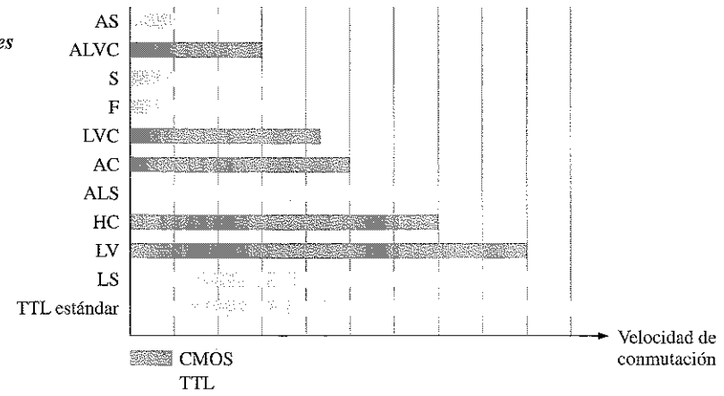
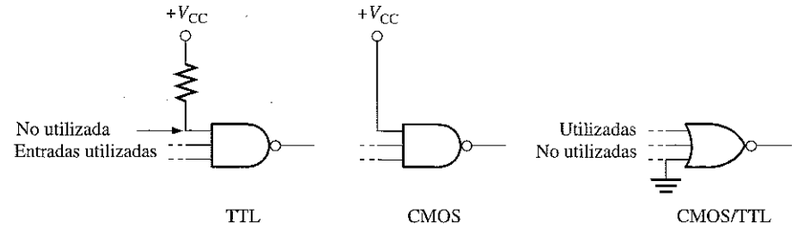


FIGURA 3.56
Comparación relativa de las velocidades de conmutación para CMOS y TTL.



Consejos Prácticos

Las entradas de puerta no utilizadas para TTL y CMOS deberían conectarse al nivel lógico apropiado (ALTO o BAJO). Para las puertas AND y NAND, es recomendable que las entradas no utilizadas se conecten a V_{CC} (a través de una resistencia de 1,0 kΩ para TTL y para las puertas OR y NOR, las puertas no utilizadas deberían conectarse a tierra.



Hojas de características

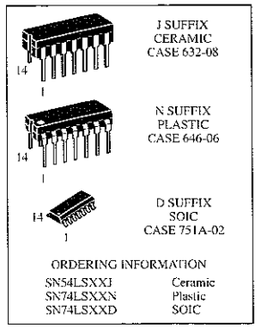
Una hoja de características típica consta de una página de información que muestra, entre otras cosas, el diagrama lógico y los encapsulados, las condiciones de operación recomendadas, las características eléctricas y las características de conmutación. En las Figuras 3.57 y 3.58 se muestran, respectivamente, parte de las hojas de características para un 74LS00 y un 74HC00A. La longitud de las hojas de características varía y unas aportan mucha más información que otras. En el Apéndice A se proporcionan algunas hojas de características adicionales.

QUAD 2-INPUT NAND GATE

• ESD > 3500 Volts

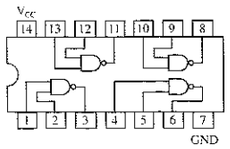
SN54/74LS00

QUAD 2-INPUT NAND GATE
LOW POWER SCHOTTKY



ORDERING INFORMATION

SN54LSXXJ	Ceramic
SN74LSXXN	Plastic
SN74LSXXD	SOIC



SN54/74LS00

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

Symbol	Parameter		Limits Min	Typ	Max	Unit	Test Conditions
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage for All Inputs
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage for All Inputs
		74			0.8		
V_{IK}	Input Clamp Diode Voltage			–0.65	–1.5	V	V_{CC} = MIN, I_{IN} = –18 mA
V_{OH}	Ouput HIGH Voltage	54	2.5	3.5		V	V_{CC} = MIN, I_{OH} = MAX, V_{IN} = V_{IH} or V_{IL} per Truth Table
		74	2.7	3.5		V	
V_{OL}	Ouput LOW Voltage	54, 74		0.25	0.4	V	I_{OL} = 4.0 mA; V_{CC} = V_{CC} MIN, V_{IN} = V_{IL} or V_{IH} per Truth Table
		74		0.35	0.5	V	I_{OL} = 8.0 mA
I_{IH}	Input HIGH Current				20	µA	V_{CC} = MAX, V_{IN} = 2.7 V
					0.1	mA	V_{CC} = MAX, V_{IN} = 7.0 V
I_{IL}	Input LOW Current				–0.4	mA	V_{CC} = MAX, I_N = 0.4 V
I_{OS}	Short Circuit Current (Note 1)		–20		–100	mA	V_{CC} = MAX
I_{CC}	Power Supply Current Total, Output HIGH				1.6	mA	V_{CC} = MAX
	Total, Output LOW				4.4		

NOTE 1: Not more than one output should be shorted at a time, nor for more than 1 second.

AC CHARACTERISTICS (T_A = 25°C)

Symbol	Parameter	Limits Min	Typ	Max	Unit	Test Conditions
t_{PLH}	Turn-Off Delay, Input to Output		9.0	15	ns	V_{CC} = 5.0 V
t_{PHL}	Turn-On Delay, Input to Output		10	15	ns	C_L = 15 pF

GUARANTEED OPERATING RANGES

Symbol	Parameter		Min	Typ	Max	Unit
V_{CC}	Supply Voltage	54	4.5	5.0	5.5	V
		74	4.75	5.0	5.25	
T_A	Operating Ambient Temperature Range	54	–55	25	125	°C
		74	0	25	70	
I_{OH}	Output Current — High	54, 74			–0.4	mA
I_{OL}	Output Current — Low	54			4.0	mA
		74			8.0	

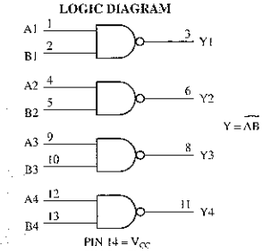
FIGURA 3.57
Una parte de la hoja de características de un 74LS00.

Quad 2-Input NAND Gate High-Performance Silicon–Gate CMOS

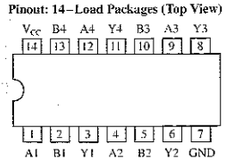
The MC54/74HC00A is identical in pinout to the LS00. The device inputs are compatible with Standard CMOS outputs; with pullup resistors, they are compatible with LSTTL outputs.

- Output Drive Capability: 10 LSTTL Loads
- Outputs Directly Interface to CMOS, NMOS and TTL
- Operating Voltage Range: 2 to 6 V
- Low Input Current: 1 µA
- High Noise Immunity Characteristic of CMOS Devices
- In Compliance With the JEDEC Standard No. 7A Requirements
- Chip Complexity: 32 FETs or 8 Equivalent Gates

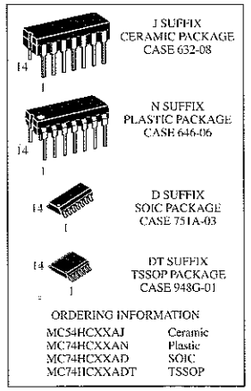
LOGIC DIAGRAM



Pinout: 14–Load Packages (Top View)



MC54/74HC00A



ORDERING INFORMATION

MC54HCXXAJ	Ceramic
MC74HCXXAN	Plastic
MC74HCXXAD	SOIC
MC74HCXXADT	TSSOP

FUNCTION TABLE

Inputs A	Inputs B	Output Y
L	L	H
L	H	H
H	L	H
H	H	L

MAXIMUM RATINGS*

Symbol	Parameter	Value	Unit
V_{CC}	DC Supply Voltage (Referenced to GND)	–0.5 to + 7.0	V
V_{in}	DC Input Voltage (Referenced to GND)	–0.5 to V_{CC} + 0.5	V
V_{out}	DC Output Voltage (Referenced to GND)	–0.5 to V_{CC} + 0.5	V
I_{in}	DC Input Current, per Pin	± 20	mA
I_{out}	DC Output Current, per Pin	± 25	mA
I_{CC}	DC Supply Current, V_{CC} and GND Pins	± 50	mA
P_D	Power Dissipation in Still Air, Plastic or Ceramic DIP†	750	mW
	SOIC Package†	500	
	TSSOP Package†	450	
T_{stg}	Storage Temperature	–65 to + 150	°C
T_L	Lead Temperature, 1 mm from Case for 10 Seconds		°C
	Plastic DIP, SOIC or TSSOP Package	260	
	Ceramic DIP	300	

* Maximum Ratings are those values beyond which damage to the device may occur. Functional operation should be restricted to the Recommended Operating Conditions.
† Derating — Plastic DIP: – 10 mW/°C from 65° to 125° C
Ceramic DIP: – 10 mW/°C from 100° to 125° C
SOIC Package: – 7 mW/°C from 65° to 125° C
TSSOP Package: – 6.1 mW/°C from 65° to 125° C

RECOMMENDED OPERATING CONDITIONS

Symbol	Parameter		Min	Max	Unit
V_{CC}	DC Supply Voltage (Referenced to GND)		2.0	6.0	V
V_{in}, V_{out}	DC Input Voltage, Output Voltage (Referenced to GND)		0	V_{CC}	V
T_A	Operating Temperature, All Package Types		–55	+125	°C
t_r, t_f	Input Rise and Fall Time	V_{CC} = 2.0 V	0	1000	ns
		V_{CC} = 4.5 V	0	500	
		V_{CC} = 6.0 V	0	400	

DC CHARACTERISTICS (Voltages Referenced to GND) MC54/74HC00A

Symbol	Parameter	Condition	V_{CC} V	Guaranteed Limit –55 to 25°C	≤85°C	≤125°C	Unit
V_{IH}	Minimum High-Level Input Voltage	V_{out} = 0.1V or V_{CC} – 0.1V; $\|I_{out}\| \le 20\mu A$	2.0	1.50	1.50	1.50	V
			3.0	2.10	2.10	2.10	
			4.5	3.15	3.15	3.15	
			6.0	4.20	4.20	4.20	
V_{IL}	Maximum Low-Level Input Voltage	V_{out} = 0.1V or V_{CC} – 0.1V; $\|I_{out}\| \le 20\mu A$	2.0	0.50	0.50	0.50	V
			3.0	0.90	0.90	0.90	
			4.5	1.35	1.35	1.35	
			6.0	1.80	1.80	1.80	
V_{OH}	Minimum High-Level Output Voltage	V_{in} = V_{IH} or V_{IL}; $\|I_{out}\| \le 20\mu A$	2.0	1.9	1.9	1.9	V
			4.5	4.4	4.4	4.4	
			6.0	5.9	5.9	5.9	
		V_{in} = V_{IH} or V_{IL}; $\|I_{out}\| \le 2.4mA$	3.0	2.48	2.34	2.20	
		$\|I_{out}\| \le 4.0mA$	4.5	3.98	3.84	3.70	
		$\|I_{out}\| \le 5.2mA$	6.0	5.48	5.34	5.20	
V_{OL}	Maximum Low-Level Output Voltage	V_{in} = V_{IH} or V_{IL}; $\|I_{out}\| \le 20\mu A$	2.0	0.1	0.1	0.1	V
			4.5	0.1	0.1	0.1	
			6.0	0.1	0.1	0.1	
		V_{in} = V_{IH} or V_{IL}; $\|I_{out}\| \le 2.4mA$	3.0	0.26	0.33	0.40	
		$\|I_{out}\| \le 4.0mA$	4.5	0.26	0.33	0.40	
		$\|I_{out}\| \le 5.2mA$	6.0	0.26	0.33	0.40	
I_{in}	Maximum Input Leakage Current	V_{in} = V_{CC} or GND	6.0	±0.1	±1.0	±1.0	µA
I_{CC}	Maximum Quiescent Supply Current (per Package)	V_{in} = V_{CC} or GND; I_{out} = 0µA	6.0	1.0	10	40	µA

AC CHARACTERISTICS (C_L = 50 pF, Input t_r = t_f = 6 ns)

Symbol	Parameter	V_{CC} V	Guaranteed Limit –55 to 25°C	≤85°C	≤125°C	Unit
t_{PLH}, t_{PHL}	Maximum Propagation Delay, Input A or B to Output Y	2.0	75	95	110	ns
		3.0	30	40	55	
		4.5	15	19	22	
		6.0	13	16	19	
t_{TLH}, t_{THL}	Maximum Output Transition Time, Any Output	2.0	75	95	110	ns
		3.0	27	32	36	
		4.5	15	19	22	
		6.0	13	16	19	
C_{in}	Maximum Input Capacitance		10	10	10	pF

Symbol	Parameter	Typical @ 25°C, V_{CC} = 5.0 V, V_{EE} = 0 V	Unit
C_{PD}	Power Dissipation Capacitance (Per Buffer)	22	pF

FIGURA 3.58
Una parte de la hoja de características de un 74HC00A.

REVISIÓN DE LA SECCIÓN 3.7

1. Enumerar tres tipos de tecnologías de circuitos integrados y nombre las dos más ampliamente utilizadas.
2. Identificar las siguientes designaciones lógicas de circuitos integrados:
   (a) LS **(b)** ALS **(c)** F **(d)** HC
   (e) AC **(f)** HCT **(g)** LV
3. Identificar los siguientes dispositivos de acuerdo a su función lógica:
   (a) 74LS04 **(b)** 74HC00 **(c)** 74LV08 **(d)** 74ALS10
   (e) 7432 **(f)** 74ACT11 **(g)** 74AHC02
4. Generalmente, ¿qué tecnología de circuitos integrados tiene la disipación de potencia más baja? ¿Y la velocidad más alta?
5. ¿Qué quiere decir el término *inversor séxtuple*? ¿Y el término *cuádruple NAND de 2 entradas*?
6. Se aplica un pulso positivo a una entrada inversora. El tiempo desde el flanco de subida de la entrada hasta el flanco de subida de la salida es 10 ns. El tiempo desde el flanco de bajada de la entrada hasta el flanco de bajada de la salida es de 8 ns. ¿Cuáles son los valores de t_{PLH} y t_{PHL}?
7. Una determinada puerta tiene un retardo de propagación de 6 ns y una disipación de potencia de 3 mW. Determinar el producto velocidad-potencia.
8. Definir I_{CCL} e I_{CCH}
9. Definir V_{IL} y V_{IH}
10. Definir V_{OL} y V_{OH}

En este punto, puede estudiar total o parcialmente el Capítulo 15, "Tecnologías de circuitos integrados". El estudio de este capítulo no es necesario para abordar cualquier otra parte de este texto y su omisión total o parcial no afectará al resto de los temas.

3.8 ■ LOCALIZACIÓN DE AVERÍAS

La localización de averías es el proceso de reconocer, aislar y corregir un fallo en un sistema o circuito. Para poder localizar las averías de forma efectiva, se debe entender cómo se supone que trabaja el circuito o sistema y se debe estar en disposición de reconocer un funcionamiento incorrecto. Por ejemplo, para determinar si una puerta lógica tiene un fallo, se debe saber cuál debe ser la salida para unas entradas dadas. En este punto, puede ser útil revisar la Sección 1.6, en la que se han tratado los pulsadores lógicos, sondas y osciloscopios.

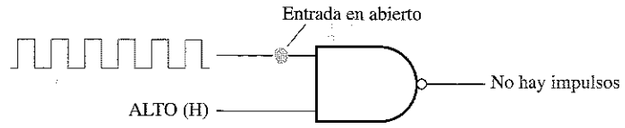
Al finalizar esta sección, el lector deberá ser capaz de:

■ Comprobar la existencia de entradas y salidas abiertas internamente en el CI, utilizando la sonda y el pulsador lógicos. ■ Reconocer los efectos de una entrada o una salida del CI cortocircuitada. ■ Detectar en una tarjeta de circuito impreso la existencia de fallos externos. ■ Localizar las averías en un sencillo contador de frecuencia utilizando un osciloscopio, además del pulsador y de la sonda lógicos.

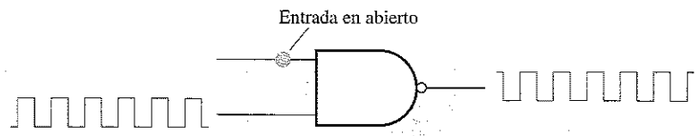
Fallos internos en los CI lógicos

Los circuitos abiertos y los cortocircuitos son los fallos más comunes en las puertas internas del CI. Se pueden producir tanto en las entradas como en las salidas de las puertas. *Antes de intentar solucionar cualquier avería, compruebe que la alimentación continua y la masa son correctas.*

Efectos de una entrada que se encuentra en circuito abierto internamente. Un circuito abierto interno es el resultado de un componente en circuito abierto o de una ruptura en la conexión entre el chip y el pin del encapsulado. Una entrada en circuito abierto impide que una señal de impulsos en esta entrada dé lugar a una salida, como se muestra en la Figura 3.59(a) para la puerta NAND de 2 entradas. Una entrada TTL en abierto actúa como un nivel ALTO, por lo que los impulsos aplicados a la entrada que está en buen estado pasan a través de la puerta NAND hasta la salida, como se muestra en la Figura 3.59(b).



(a) Si se aplican impulsos en una entrada en abierto no se generan impulsos a la salida.



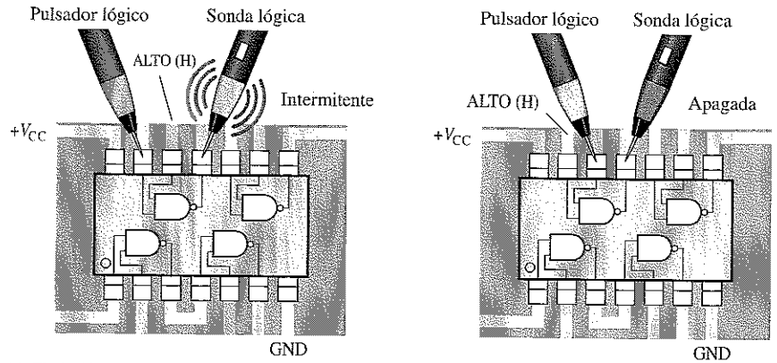
(b) Si se aplican impulsos en la entrada que está bien, se generarán impulsos a la salida en las puertas TTL NAND y AND, debido a que la entrada en abierto actúa como un nivel alto. Esto no se cumple para CMOS.

FIGURA 3.59
Efecto de una entrada en circuito abierto en una puerta NAND.

Condiciones para probar las puertas. Cuando se prueba una puerta NAND o una puerta AND, debe asegurarse siempre de que la entrada a la que no se aplican impulsos se encuentra a nivel ALTO, para activar la puerta. Cuando se prueba una puerta NOR o una puerta OR, debe asegurarse siempre de que la entrada a la que no se aplican impulsos se encuentra a nivel BAJO. Cuando se prueba una puerta XOR o XNOR, el nivel de la entrada a la que no se aplican impulsos no importa, ya que los impulsos aplicados en la otra entrada forzarán a que las entradas se encuentren, alternativamente, en el mismo nivel o en niveles opuestos.

Localización de fallo: una entrada en circuito abierto. La localización de este tipo de fallo es en la mayoría de los casos muy fácil utilizando un pulsador y una son-

da lógicos, como se muestra en la Figura 3.60, para el caso de una puerta NAND de 2 entradas.



(a) El pin 13 de entrada y el pin 11 de salida están bien. (b) El pin 12 de entrada está en abierto.

FIGURA 3.60
Localización de averías en una puerta NAND con una entrada en circuito abierto, empleando una sonda y un pulsador lógicos.

El primer paso en la localización de averías de un CI, cuando se sospecha que está fallando, es asegurarse de que la tensión de alimentación continua (V_{CC}) y la masa están conectadas a los pines apropiados del CI. Después, con el pulsador se aplican impulsos continuos a una de las entradas de la puerta, asegurándose de que las otras entradas están a nivel ALTO (en el caso de una puerta NAND). En la Figura 3.60(a), se comienza a aplicar los impulsos en el pin 13, ya que se ha determinado que es una de las entradas de la puerta de la que se sospecha el fallo. Si en la salida correspondiente (en este caso el pin 11) se detecta una respuesta mediante la intermitencia de la sonda, entonces el pin 13 de entrada no está en abierto. Consecuentemente, esto prueba también que la salida no está en abierto. A continuación, se prueba otra entrada de la puerta (pin 12). La luz de la sonda no se enciende, lo que indica que no hay impulsos en la salida en el pin 11, y la salida está a nivel BAJO, como muestra la Figura 3.60(b). Obsérvese que la entrada en la que no se aplican impulsos debe estar a nivel ALTO, en los casos de puertas NAND y AND. Si se tratara de una puerta NOR, la entrada en la que no se aplican impulsos debería estar a nivel BAJO.

Efectos de una salida que se encuentra internamente en circuito abierto. Una salida que esté internamente en circuito abierto impide que una señal en cualquiera de las entradas llegue hasta la salida. Por tanto, independientemente de las condiciones de entrada, la salida no se ve afectada. El nivel en el pin de salida del CI dependerá de a qué esté conectada la salida externamente, por lo que podría estar a nivel ALTO, BAJO o flotante (no fijado a ninguna referencia). En cualquier caso, una sonda lógica en el pin de salida no lucirá de forma intermitente.

Localización de fallo: salida en abierto. La Figura 3.61 ilustra la localización de una salida en abierto en una puerta NOR utilizando un pulsador y una sonda lógicos.

En la parte (a) de la figura, se aplican impulsos a una de las entradas de las que se sospecha (en este caso, el pin 11), y la sonda en la salida (pin 13) indica que no hay impulsos. En la parte (b) de la figura, se aplican impulsos a la otra entrada (pin 12) y, de nuevo, no hay indicación de impulsos en la salida. Bajo la condición de que en la entrada en la que no se aplican impulsos esté a nivel BAJO, esta prueba demuestra que la salida está internamente en circuito abierto.

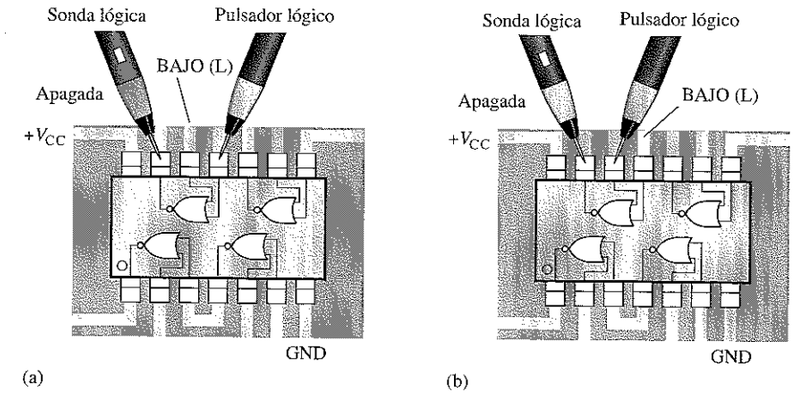


FIGURA 3.61
Localización de una salida en circuito abierto en una puerta NOR, empleando una sonda y un pulsador lógicos.

Entrada o salida cortocircuitadas. Aunque no es un fallo común como un abierto, se puede producir un cortocircuito interno a la tensión de alimentación, a masa, a otra entrada o a una salida. Cuando una entrada o una salida se cortocircuita a la alimentación, permanecerá en estado ALTO. Si una entrada o una salida se cortocircuita a masa, permanecerá a nivel BAJO (0 V). Si dos entradas o una entrada y una salida se cortocircuitan entre sí, entonces estarán siempre al mismo nivel.

Vamos a reservar el estudio de la localización de averías producidas por cortocircuitos para el Capítulo 5, en el que veremos la utilización de un indicador de corriente, para localizar un corto en situaciones en las que varias puertas están interconectadas.

Abiertos y cortos externos

Muchos de los fallos que afectan a los CI digitales se deben a fallos externos a los mismos. Se incluyen en este tipo de fallos las soldaduras incorrectas, salpicaduras de estaño, cortes de conductores, tarjetas de circuito impreso grabadas incorrectamente, y rupturas en las conexiones o en las interconexiones del circuito impreso. Estas condiciones de circuitos abiertos o cortocircuitos tienen los mismos efectos sobre las puertas lógicas que los fallos internos y, básicamente, se localizan mediante los mismos métodos. Cuando se sospecha que algo está fallando, lo primero que debe hacer un técnico es una inspección visual del circuito.

EJEMPLO 3.23 Se quiere probar un 74LS10, una triple puerta NAND de 3 entradas, que es uno de los muchos CI de una tarjeta de circuito impreso. Ha probado los pines 1 y 2 con su sonda lógica, y ambos están a nivel ALTO (H). Después, aplica el pulsador lógico al pin 13, y la sonda lógica se coloca, en primer lugar, en el pin 12 y luego en la pista del circuito impreso, como indica la Figura 3.63. Basándose en la respuesta de la sonda, ¿cuál es, probablemente, el principal problema?

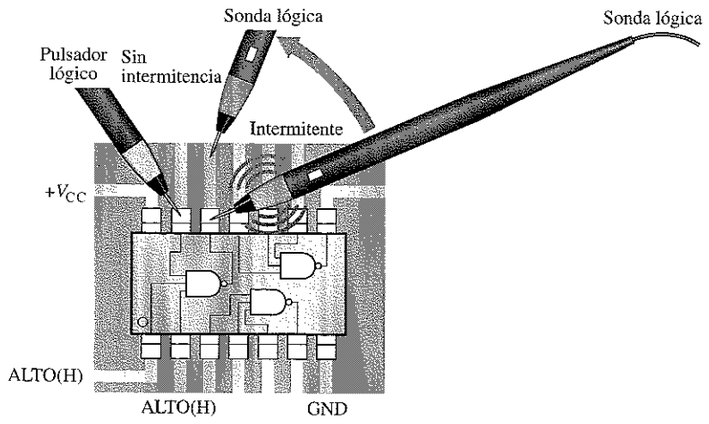


FIGURA 3.62

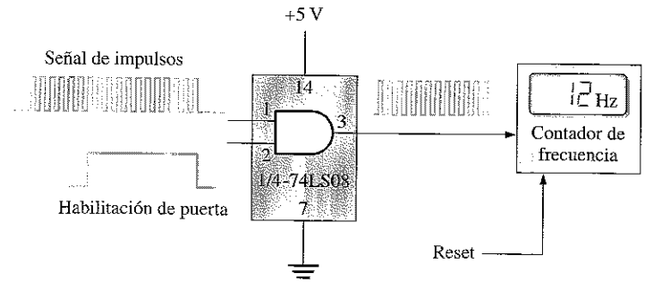
Solución. El indicador de la sonda se enciende intermitentemente, lo que demuestra que detecta impulsos en la salida de la puerta en el pin 12 pero, sin embargo, no detecta actividad en la pista de la tarjeta de circuito impreso. La puerta está trabajando correctamente, pero la señal no pasa del pin 12 del CI a la pista del circuito impreso.
Lo más probable es que haya una mala soldadura entre el pin 12 del CI y la tarjeta de circuito impreso, lo que da lugar a un circuito abierto. Por tanto, debería volver a soldar este punto y probar de nuevo.

Problema relacionado. Si la sonda no se enciende en ningún punto en la Figura 3.62, ¿qué fallo(s) indica esto?

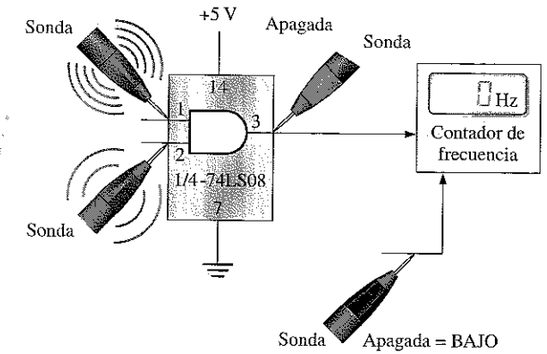
Casi siempre deberán localizarse fallos en los circuitos integrados mientras que están montados en tarjetas de circuito impreso o en prototipos ensamblados, y están interconectados con otros circuitos integrados. Según se vaya avanzado a través de este libro, aprenderemos cómo se usan diferentes tipos de CI digitales conjuntamente para realizar funciones de sistemas. Sin embargo, en este momento, vamos a concentrarnos en las puertas individuales de los circuitos digitales. Esta limitación no nos impide abordar el concepto de sistema en un nivel básico y simplificado, como ya hemos hecho varias veces.

Para continuar con el concepto de sistemas, los Ejemplos 3.24 y 3.25 se ocupan de la localización de averías en el contador de frecuencia introducido en la Sección 3.2.

EJEMPLO 3.24 Después de intentar trabajar con el contador de frecuencia de la Figura 3.63, se encuentra con que constantemente da una lectura de todo 0s en el display, independientemente de la frecuencia de entrada. Determinar la causa de este mal funcionamiento.



(a) Así es como el contador debería funcionar con una señal de entrada de 12 Hz.



(b) El display muestra una frecuencia de 0 Hz y se muestran las indicaciones de la sonda lógica.

FIGURA 3.63

Solución. Existen tres posibles causas:

1. Un nivel activo o verdadero constante en la entrada reset del contador hace que el contador esté siempre a cero.
2. No hay tren de impulsos en la entrada del contador, debido a que existe un circuito abierto o un cortocircuito en el contador. Este problema impediría que el contador avanzara después de haber sido puesto a cero.
3. No hay tren de impulsos a la entrada del contador porque hay un circuito abierto en la salida de la puerta AND o por la ausencia de señales de entrada, lo que de nuevo impide que el contador avance a partir de cero.

La Figura 3.63(a) nos proporciona un ejemplo de cómo el contador de frecuencia debería trabajar con un tren de impulsos a 12 Hz aplicado a la entrada de la puerta AND. La parte (b) de la figura muestra que el display indica, incorrectamente, 0 Hz.

El primer paso es asegurarse de que V_{CC} y masa están correctamente conectadas en todos los puntos; supongamos que están bien. A continuación, probamos aplicando impulsos en ambas entradas de la puerta AND. La sonda indica que hay impulsos en ambas entradas. Se aplica la sonda a la entrada reset del contador y muestra un nivel BAJO, que corresponde a un nivel no verdadero (supongamos que esta información se ha encontrado en la hoja de características del contador) y, por tanto, aquí no está el problema. La siguiente prueba con la sonda se hace en el pin 3 del 74LS08, y muestra que no hay impulsos en la salida de la puerta AND, lo que indica que la salida de la puerta está internamente en circuito abierto. Debe reemplazar el CI 74LS08 y probar de nuevo el funcionamiento.

Problema relacionado. Si el pin 2 de la puerta AND 74LS08 estuviese en circuito abierto, ¿qué indicación vería en el display?

EJEMPLO 3.25 El contador de frecuencia mostrado en la Figura 3.64 parece medir incorrectamente la frecuencia de las señales de entrada. Se determina que, cuando se aplica al pin 1 de la puerta AND una señal con una frecuencia conocida, el display indica una frecuencia más alta. Determinar dónde está el fallo. Las lecturas de la pantalla están expresadas en segundos/división.

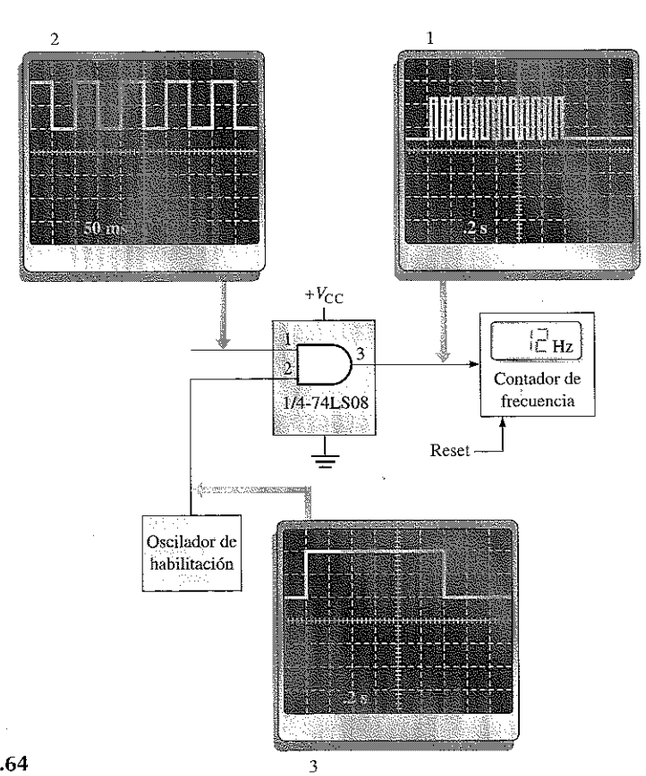


FIGURA 3.64

Solución. Recuérdese de la Sección 3.2 que los impulsos de entrada pueden pasar a través de la puerta AND exactamente durante 1 s. El número de impulsos que se cuentan en 1 s es igual a la frecuencia en hertzios (ciclos por segundo). Por tanto, el intervalo de 1 s, que es generado por el impulso de habilitación aplicado en el pin 2 de la puerta AND, es crítico para obtener una medida de frecuencia precisa. Los impulsos de habilitación se generan internamente mediante un circuito oscilador de precisión. El ancho del impulso debe ser exactamente de 1 s y, en este caso, se produce cada 3 s para actualizar el contador. Justo después de cada impulso de habilitación, el contador se pone a cero, por lo que empieza a contar de nuevo.

Puesto que el contador cuenta más impulsos que los que debería, dando lugar a que se obtenga una lectura de frecuencia más alta, parece que el impulso de activación es el sospechoso principal. La medida exacta del intervalo de tiempo, en este caso, se lleva a cabo con un osciloscopio en lugar de con una sonda lógica. La sonda lógica sólo indica la presencia de impulsos, luego no sirve para la medida de tiempos o frecuencias.

Se aplica al pin 1 de la puerta AND un tren de impulsos de entrada de exactamente 10 Hz e, incorrectamente, el display presenta 12 Hz. La primera medida en el osciloscopio, en la salida de la puerta AND, muestra que hay 12 impulsos por cada impulso de activación. La segunda medida del osciloscopio verifica que la frecuencia de entrada es exactamente 10 Hz (periodo = 100 ms). La tercera medida del osciloscopio facilita el ancho del impulso de activación, que es 1,2 s, mayor por tanto que 1 s.

La conclusión es que el circuito oscilador que genera el impulso de activación no está bien calibrado por alguna razón, por lo que debe repararse o reemplazarse.

Problema relacionado. ¿De qué se sospecharía si la lectura indicara una frecuencia menor de lo que debe ser?

Consejos Prácticos

La correcta puesta a tierra es muy importante cuando se prepara un circuito para tomar medidas o trabajar en él. La correcta puesta a tierra del osciloscopio protege al usuario frente a descargas eléctricas y la conexión a tierra del propio usuario protege los circuitos frente a posibles daños. Poner a tierra el osciloscopio quiere decir conectarlo a la toma de masa, introduciendo el conector de tres tomas en un enchufe con toma de masa. Como ya sabrá, ponerse a tierra el usuario significa que debe utilizarse una pulsera de conexión a masa, especialmente cuando se trabaja con circuitos CMOS.

Para obtener medidas precisas, también debe asegurarse de que la tierra del circuito que se está probando sea la misma que la del osciloscopio. Esto puede hacerse conectando el terminal de tierra de la sonda del osciloscopio a un punto de tierra conocido del circuito, como puede ser el chasis metálico o un punto de tierra en la tarjeta de circuito impreso. También se puede conectar la tierra del circuito al conector GND ubicado en el panel frontal del osciloscopio.

REVISIÓN DE LA SECCIÓN 3.8

1. ¿Cuáles son los tipos de fallos más comunes en los circuitos integrados?
2. Si se aplican dos señales de entrada diferentes a una puerta NAND TTL de 2 entradas y la señal de salida es como una de las entradas, pero invertida, ¿cuál es el problema más probable?
3. Citar una ventaja del osciloscopio frente a la sonda lógica.
4. Citar una ventaja de la sonda lógica frente al osciloscopio.

3.9 ■ APLICACIÓN A LOS SISTEMAS DIGITALES

El sistema de control y recuento de pastillas que se ha estudiado en los Capítulos 1 y 2, va a ser de nuevo objeto de esta aplicación a los sistemas digitales. Si se desea, se pueden volver a ver dichas secciones antes de comenzar con ésta. En esta sección, nos centraremos en la aplicación de las puertas lógicas a determinadas funciones del sistema.

Al finalizar esta sección, el lector deberá ser capaz de:

- Implementar la función comparador con las puertas básicas vistas en este capítulo.
- Explicar cómo se pueden usar una puerta AND y un inversor para resolver determinados problemas de temporización en el sistema.
- Utilizar un diagrama de tiempos para analizar la lógica del comparador y el funcionamiento del sistema.
- Localizar las averías del comparador implementado, utilizando un pulsador y una sonda lógicos.

La función del comparador

En este sistema, el **comparador** detecta cuándo el número binario contenido en el contador, que es el número actual de pastillas introducidas en el bote, es igual al número binario preseleccionado, que representa el número máximo de pastillas por bote. Cuando estos dos números son iguales, el bote está lleno, y contiene el número de pastillas preseleccionado, por lo que la válvula deberá cerrarse para cortar el flujo de pastillas al interior del bote.

Aunque existen CI diseñados específicamente como comparadores, que estudiaremos más adelante, este comparador se ha implementado utilizando puertas individuales para estar dentro del ámbito de este capítulo. Recuérdese que la salida de una puerta OR-exclusiva se pone a nivel BAJO (0) cuando ambos bits de entrada son iguales, como muestra la Figura 3.65. Esta característica de la puerta OR-exclusiva (XOR) se puede usar ventajosamente en la aplicación del comparador, ya que estamos buscando la aparición de dos números binarios iguales.

FIGURA 3.65
La salida de una puerta XOR es 0 cuando sus entradas son iguales.

En esta aplicación, el número de pastillas introducido en el teclado numérico se presenta a la salida del convertidor de código BCD-binario como un número binario fijo de 8 bits. El contador binario también contiene un número de 8 bits, que representa el número real de pastillas que hay en el bote en un determinado instante, y este número aumenta cada vez que se cuenta una pastilla. Estos dos números binarios de 8 bits se aplican a las entradas del comparador, que se muestra en la Figura 3.66.

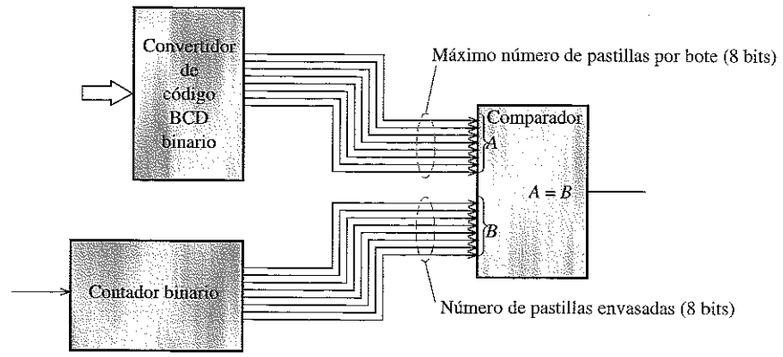


FIGURA 3.66
Diagrama de bloques de la parte del sistema que contiene el comparador, el contador y el convertidor de código.

Implementación del comparador

Una puerta XOR puede admitir dos bits en sus entradas, e indicar en su salida si los dos bits son iguales o diferentes. Para comparar los dos números de 8 bits, se requieren ocho puertas XOR. Cada puerta compara los bits que le corresponden de cada uno de los números, es decir, una puerta compara los LSB de los dos números; la siguiente puerta compara los dos siguientes bits, y así sucesivamente. Por último, la puerta ocho compara los MSB de los dos números. Una puerta negativa-AND (NOR) de 8 entradas se usa entonces para detectar cuándo todas las salidas de las puertas XOR están a nivel BAJO, lo que indica que los bits son iguales. El esquema lógico básico se muestra en la Figura 3.67(a).

No existe un CI que implemente una puerta NOR de 8 entradas para aplicar la puerta negativa-AND requerida; por tanto, se debe usar una puerta NAND de 8 entradas. Puesto que una puerta XOR genera un nivel BAJO (0) cuando sus entradas son iguales (1 o 0), se requiere un inversor en la salida de cada puerta XOR para generar un nivel ALTO para la puerta NAND.

La Figura 3.67(b) muestra la implementación real utilizando dos 74LS86, cuádruple puerta XOR, dos inversores séxtuples 74LS04, y una puerta NAND de 8 entradas, el 74LS30. Esta implementación real es la misma, en términos lógicos, que la implementación básica mostrada en la parte (a) de la figura. Cuando los dos números de entrada de 8 bits son iguales, la salida de la puerta NAND es un nivel BAJO. La salida de esta puerta debe invertirse para proporcionar el nivel ALTO requerido para controlar la válvula y otros circuitos. Existe un método para implementar el comparador utilizando menos dispositivos, pero esto queda reservado para las actividades prácticas de sistemas.

Un problema en el sistema

La salida a nivel ALTO del comparador hace que la válvula del embudo de pastillas se cierre y corte el flujo, lo que da lugar a que la cinta transportadora mueva el siguiente

bote a la posición adecuada, y hace que la salida del sumador se almacene en el registro B (véase la Figura 2.6).

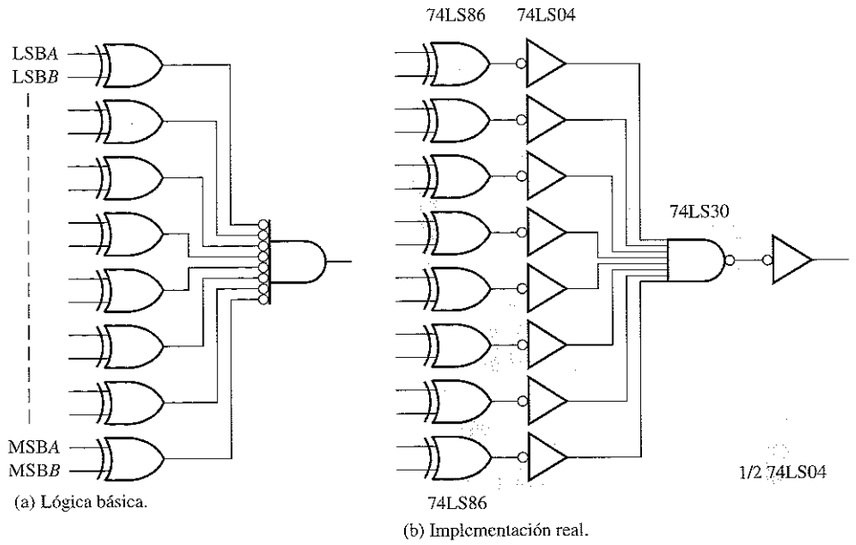


FIGURA 3.67
Implementación de la lógica del comparador.

Existe un problema de temporización cuando la salida del comparador pasa a nivel ALTO. Cuando pasa la última pastilla por el sensor, y el contador se incrementa igualándose al número preseleccionado de pastillas, existe la posibilidad de que otra pastilla haya pasado ya por la válvula y haga que el contador se incremente de nuevo, haciendo que la salida del comparador pase otra vez a nivel BAJO.

Solución. Para corregir este problema de temporización, se usan el inversor y la puerta AND mostrados en la Figura 3.68, de modo que se inhiba el impulso del sensor al contador tan pronto como los números binarios sean iguales. Mientras que la salida del comparador esté a nivel BAJO, el nivel ALTO en la salida del inversor activa a la puerta AND y permite que los impulsos del sensor pasen a través de la puerta hasta el contador. Cuando la salida del comparador pasa a nivel ALTO, la salida del inversor pasa a nivel BAJO y evita que otro impulso del sensor llegue a la salida de la puerta AND, impidiendo que el contador avance si el sensor ha detectado una pastilla más.

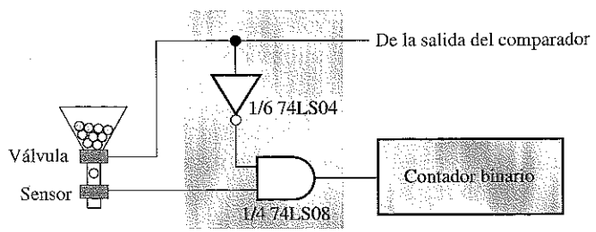


FIGURA 3.68
Diagrama parcial del sistema que muestra la lógica de inhibición del contador.

■ PRÁCTICAS DE SISTEMAS

Análisis y diseño

Se supone que el sistema funciona correctamente. La lógica del comparador y la lógica de inhibición del contador se han implementado en la zona de la tarjeta del sistema que se muestra en la Práctica de sistemas 1. La lógica del comparador está formada por dos cuádruples puertas XOR, 74LS86; dos inversores séxtuples 74LS04 y una puerta NAND de 8 entradas, el 74LS30. El circuito de inhibición del contador utiliza uno de los inversores de la izquierda del 74LS04 y una de las puertas AND de una cuádruple puerta AND 74LS08. Este diseño se simplificará en la actividad número 6 de la Práctica de sistemas 1.

Las líneas oscuras en la tarjeta del sistema conectan puntos correspondientes de la cara posterior de la tarjeta a través del metalizado de cada taladro. Las conexiones por la cara posterior para V_{CC} y tierra (GND) no se muestran.

Realizar las actividades que se enumeran en la Práctica de sistemas 1.

Localización de averías

Observación. Se produce un problema en el sistema, que consiste en que las pastillas continúan cayendo después de que el bote está ya lleno, y los botes colocados sobre la cinta transportadora no avanzan.

Conclusión. Parecen existir dos problemas: (1) la válvula no se cierra, y (2) la cinta transportadora no avanza. Sin embargo, es improbable que estos dos elementos fallen a la vez, por lo que debemos buscar una causa común y suponer que la válvula y la cinta no fallan. ¿Qué controla a la válvula y al avance de la cinta transportadora? Cuando la salida del comparador pasa a nivel ALTO, la válvula se debería cerrar y la cinta se debería mover al bote siguiente para colocarlo. Puede ser que algo impida que la salida del comparador pase a nivel ALTO.

Verificar la conclusión. Si la salida del comparador permanece a nivel BAJO, la conclusión es correcta. Si pasa a nivel ALTO, quizás la válvula y la cinta fallan y la conclusión es errónea. Supongamos que la salida del comparador está siempre a nivel BAJO y continuemos.

Prioridad número uno. La primera prioridad después de haber determinado que hay un fallo en la tarjeta de circuito del sistema y no en la válvula o en el mecanismo de la cinta transportadora, es minimizar las pérdidas de tiempo del sistema (pérdidas de tiempo significan pérdidas de dinero). En primer lugar, hay que comprobar que el sensor está generando los impulsos que llegan al contador, y que el sistema de alimentación funciona. Suponiendo que esto está bien, se debe apagar la alimentación del sistema y extraer la tarjeta del sistema. Insertar una tarjeta de repuesto que funcione (suponemos que siempre se tiene un recambio disponible). Conectar la alimentación e inicializar el sistema. Suponemos que ahora funciona correctamente.

El siguiente paso es encontrar el problema que hace fallar a la tarjeta y repararlo, para disponer de una tarjeta de recambio cuando se produzca otro problema.

Posibles fallos. El siguiente paso es enumerar los fallos en la lógica del comparador y en la lógica de inhibición del contador, y los fallos externos al circuito lógico, que podrían mantener a nivel BAJO la salida del comparador cuando debería estar a nivel ALTO. Esta lista está referida al esquema de la Práctica de sistemas 2.

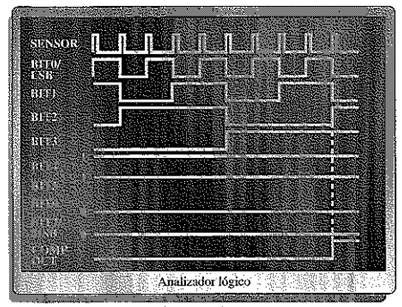
1. El contador binario no funciona.
2. La entrada o la salida del inversor 9 está en circuito abierto, o la salida está cortocircuitada.
3. La salida del inversor 10 está en circuito abierto o la salida está cortocircuitada.
4. La salida de la puerta NAND está en circuito abierto.
5. Hay una entrada en circuito abierto o una salida cortocircuitada en uno de los inversores de 1 a 8.
6. Hay una salida en circuito abierto en una de las puertas XOR.
7. Una entrada de una de las puertas XOR está en circuito abierto o cortocircuitada, de tal forma que impide que se produzca la comparación.
8. La entrada del sensor a la puerta AND está en abierto.
9. La salida de la puerta AND está en circuito abierto.

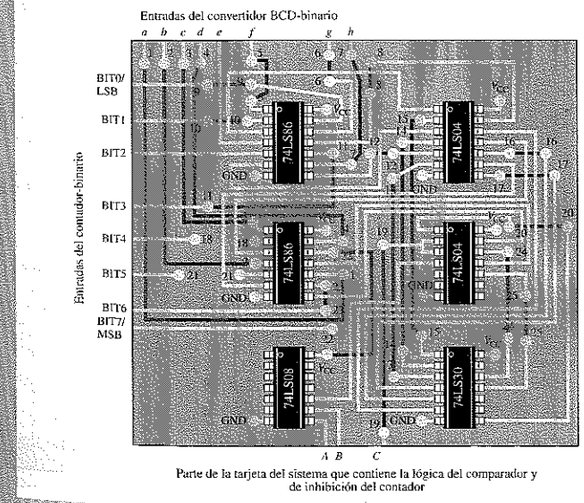
Aislamiento de los elementos que fallan. Colocada la tarjeta del sistema que falla en el banco de trabajo, se conectan la masa y la alimentación, y se aplica un impulso a la entrada del sensor. Se puede utilizar un multímetro digital para asegurarse de que V_{CC} y masa están conectadas a cada CI, y se puede usar una sonda lógica o un osciloscopio para comprobar el contador. Suponiendo que las conexiones a la alimentación y a masa están bien, y que el contador parece que está funcionando, se comprueban los circuitos lógicos del comparador y el circuito de inhibición del contador, mediante el pulsador y la sonda lógicos, como se muestra en las diversas medidas indicadas en el apartado Práctica de sistemas 2. Aunque la probabilidad de que sólo exista un fallo es alta, se presentan varias posibilidades para que adquiera práctica en llegar a una conclusión basada en un conjunto de medidas. Tenga en mente que no se incluyen todos los posibles fallos en la lista de actividades de la Práctica de sistemas 2.

■ PRÁCTICAS DE SISTEMAS 1: Análisis y diseño

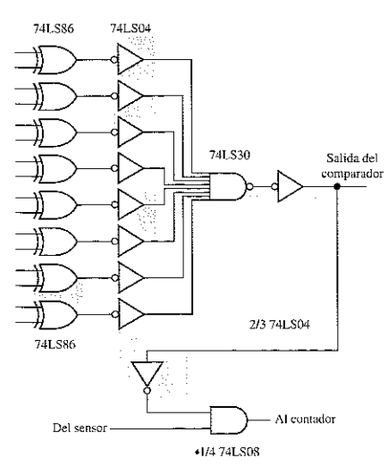
- ***Actividad 1***. Comprobar esta parte de la tarjeta del sistema con el esquema, para verificar que ambos se corresponden.
- ***Actividad 2***. La pantalla del analizador lógico muestra parte del diagrama de tiempos de las señales procedentes del contador que va a las entradas del comparador, y la salida del comparador. Determinar cuántas pastillas se introducen en cada bote.
- ***Actividad 3***. Determinar el valor presente (que procede del convertidor de código BCD-binario) en cada una de las entradas (*a, b, c,* etc.) en esta parte de la tarjeta del sistema.
- ***Actividad 4***. Identificar las tres líneas etiquetadas como *A*, *B* y *C* situadas en la parte inferior de la tarjeta utilizando el esquema como referencia.
- ***Actividad 5***. Verificar que las lecturas en el DMM (multímetro digital) cumplen las especificaciones de la hoja de características en cada uno de los puntos de la tarjeta del sistema en los que se está midiendo.
- ***Actividad 6***. La lógica del comparador y la lógica de inhibición del contador se pueden simplificar. Modificar el circuito para reducir el número de circuitos integrados sin variar la operación lógica global. Este rediseño requerirá sustituir algunos de los tipos de puertas lógicas utilizados por otros. Para determinar los CI apropiados para este rediseño puede recurrir a un catálogo de circuitos integrados.

La lógica de la tarjeta de circuito impreso del comparador se encuentra en el archivo SA03.EWB del CD-ROM EWB. Utilizar el Electronics Workbench para simular una prueba de este circuito.



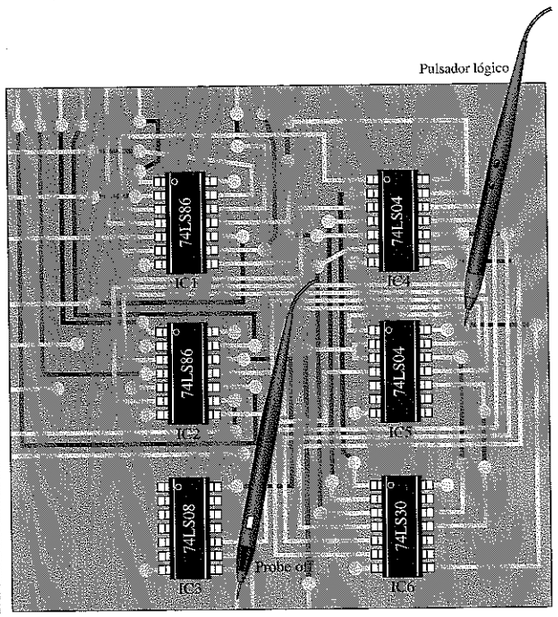


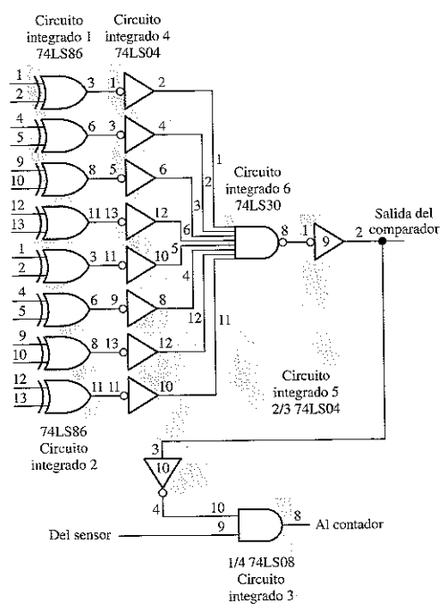
Parte de la tarjeta del sistema que contiene la lógica del comparador y de inhibición del contador



■ PRÁCTICAS DE SISTEMAS 2: Localización de averías

- ***Actividad 1***. Como se puede ver, se aplican un pulsador y una sonda lógicos a la tarjeta del sistema. La luz de la sonda está apagada, lo que indica que no se detectan impulsos. Determinar si esta indicación establece que hay o no un fallo. Si existe un fallo, especificar el dispositivo o dispositivos que fallan. Si no se puede aislar el fallo (suponiendo que existe) en un único circuito integrado, explicar por qué y establecer qué otras pruebas hay que llevar a cabo.
- ***Actividad 2***. En la tabla se indican siete conjuntos *independientes* de medidas del pulsador/sonda, correspondientes a la parte de la tarjeta que contiene el comparador y la lógica de inhibición del contador. Para cada conjunto de medidas, determinar si indican que existe un fallo o no. Si indican fallo, especificar el fallo y cómo se repara. Si el conjunto de medidas no lleva a ninguna conclusión, explicar por qué.





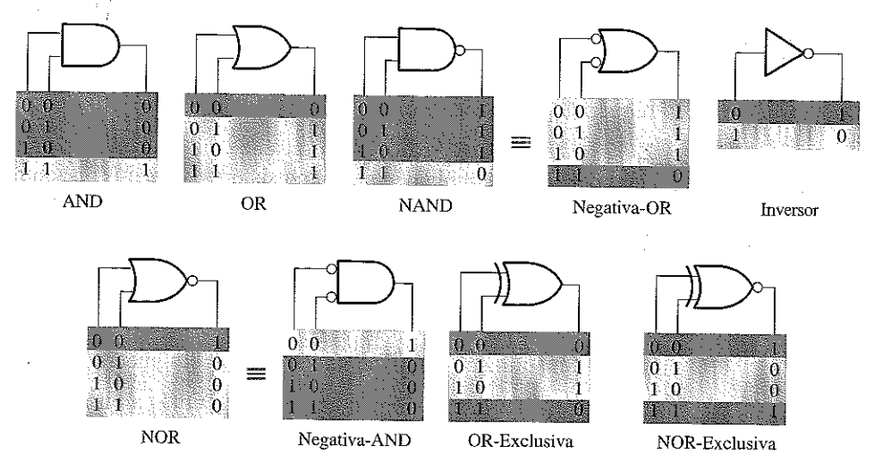
	IC#	Pin #	IC#	Pin #	Indicación	IC#	Pin #	Indicación	IC#	Pin #	Indicación
Conjunto de medidas	1	1	1	3	Intermitente	4	1	Intermitente	4	2	Apagada
Conjunto de medidas	1	4	1	6	Intermitente	4	3	Apagada			
Conjunto de medidas	2	9	2	8	Apagada						
Conjunto de medidas	4	11	4	10	Intermitente	6	5	Intermitente	6	8	Encendida
Conjunto de medidas	5	1	5	2	Intermitente	5	3	Intermitente	5	4	Apagada
Conjunto de medidas	6	1	6	8	Encendida						
Conjunto de medidas	3	9	3	8	Apagada						

REVISIÓN DE LA SECCIÓN 3.9

1. En la Práctica de sistemas 1, ¿por qué BIT4, BIT5, BIT6 y BIT7 nunca cambian?
2. En la Práctica de sistemas 1, ¿cuál es la tensión máxima que debería presentar el DMM de la izquierda? ¿Y el DMM de la derecha?
3. ¿Por qué no puede probarse de forma completa la puerta NAND de la lógica del comparador con un pulsador y una sonda lógicos?

■ RESUMEN

- La salida de un inversor es el complemento de la entrada.
- La salida de una puerta AND es un nivel ALTO sólo si todas las entradas están a nivel ALTO.
- La salida de una puerta OR es un nivel ALTO si cualquiera de las entradas está a nivel ALTO.
- La salida de una puerta NAND es un nivel BAJO sólo si todas las entradas están a nivel ALTO.
- La puerta NAND puede expresarse como una puerta negativa-OR, cuya salida es un nivel ALTO cuando cualquier entrada está a nivel BAJO.
- La salida de una puerta NOR es un nivel BAJO si cualquiera de las entradas está a nivel ALTO.
- La puerta NOR puede expresarse como una puerta negativa-AND, cuya salida es un nivel ALTO sólo si todas las entradas están a nivel BAJO.
- La salida de una puerta OR-exclusiva es un nivel ALTO cuando las entradas son distintas.
- La salida de una puerta NOR-exclusiva es un nivel BAJO cuando las entradas son distintas.
- Los símbolos distintivos y tablas de verdad para los distintos tipos de puertas lógicas (sólo para dos entradas) se muestran en la Figura 3.69.



Nota: Los estados activos se muestran con una trama más oscura.

FIGURA 3.69

- Las tecnologías TTL y ECL se basan en transistores de unión bipolares.
- La tecnología CMOS se basa en los transistores de efecto de campo MOS.
- Por regla general, los dispositivos ECL son los de mayor velocidad de conmutación, y los TTL y CMOS están más o menos emparejados, aunque CMOS está rápidamente tomando ventaja.
- Por regla general, los dispositivos CMOS tienen el consumo de potencia más bajo, seguidos de los TTL, y los ECL tienen el mayor consumo de potencia.

- La disipación media de potencia de una puerta lógica es:

$$P_D = V_{CC}\left(\frac{I_{CCH} + I_{CCL}}{2}\right) \quad \textbf{(3.2)}$$

- El producto velocidad-potencia de una puerta lógica es

$$SPP = t_P P_D \quad \textbf{(3.3)}$$

■ PALABRAS CLAVE

Estos términos se encuentran también en el glosario situado al final del libro.

Álgebra booleana Las matemáticas de los circuitos lógicos.

ANSI American National Standards Institute, Instituto Nacional Americano de Normalización.

Carga unidad Una medida del fan-out. Una entrada de una puerta representa una carga unidad a la salida de otra puerta, dentro de la misma familia de circuitos integrados.

CMOS Complementary Metal-Oxide Semiconductor, metal-óxido semiconductor complementario; un tipo de circuito de transistores que utiliza transistores de efecto de campo.

Complementación Inversión. Un nivel BAJO es el complemento de un nivel ALTO y viceversa.

Disipación de potencia El producto de la tensión de alimentación continua y la corriente de alimentación continua en un circuito electrónico. Cantidad de potencia requerida por un circuito.

ECL Emitter-Coupled Logic, lógica de emisor acoplado; un tipo de circuito lógico integrado que se implementa con transistores bipolares no saturados.

Fan-out Número de entradas de puertas equivalentes de la misma familia que puede excitar una puerta lógica.

Habilitar Activar o poner en modo de operación en un circuito. Una entrada de un circuito lógico que activa su funcionamiento.

IEEE Institute of Electrical and Electronic Engineers.

Inversión La conversión de un nivel ALTO (1) a un nivel BAJO (0), o viceversa.

Multiplicación booleana En el álgebra booleana, la operación AND.

Negativa-AND Operación equivalente a una puerta NOR, en la que se obtiene una salida a nivel ALTO cuando todas sus entradas están a nivel BAJO.

Negativa-OR Operación equivalente a una puerta NAND, en la que se obtiene una salida a nivel ALTO cuando una o más entradas están a nivel BAJO.

Producto velocidad-potencia Parámetro de funcionamiento que consiste en el producto del retardo de propagación y la disipación de potencia en un circuito digital.

Puerta AND Puerta lógica que produce una salida a nivel ALTO sólo cuando todas las entradas están a nivel ALTO.

Puerta NAND Puerta lógica que produce una salida a nivel BAJO sólo si todas las entradas están a nivel ALTO.

Puerta NOR Puerta lógica en la que la salida es un nivel BAJO cuando al menos una de las entradas está a nivel ALTO.

Puerta NOR-exclusiva (XNOR) Puerta lógica que produce una salida a nivel BAJO sólo cuando las dos entradas tienen niveles opuestos.

Puerta OR Puerta lógica que produce una salida a nivel ALTO cuando una o más entradas están a nivel ALTO.

Puerta OR-exclusiva (XOR) Puerta lógica que produce una salida a nivel ALTO sólo cuando las dos entradas tienen niveles opuestos.

Retardo de propagación Intervalo de tiempo que transcurre entre la transición de entrada y su correspondiente transición de salida en un circuito lógico.

Suma booleana En el álgebra booleana, la operación OR.

Tabla de verdad Tabla que muestra las entradas y los correspondientes niveles de salida de un circuito lógico.

TTL Transistor-Transistor Logic, lógica transistor-transistor, un tipo de circuito integrado que utiliza transistores de unión bipolares.

■ AUTOTEST

1. Cuando la entrada de un inversor es un nivel ALTO (1), la salida es:
(a) un nivel ALTO o 1 **(b)** un nivel BAJO o 1
(c) un nivel ALTO o 0 **(d)** un nivel BAJO o 0

2. Un inversor realiza la operación conocida como:
(a) complementación **(b)** afirmación
(c) inversión **(d)** las respuestas (a) y (c)

3. La salida de una puerta AND con entradas *A*, *B*, y *C* está a 1 (nivel ALTO) cuando
(a) $A = 1, B = 1, C = 1$ **(b)** $A = 1, B = 0, C = 1$ **(c)** $A = 0, B = 0, C = 0$

4. La salida de una puerta OR con entradas *A*, *B*, y *C* está a 1 (nivel ALTO) cuando
(a) $A = 1, B = 1, C = 1$ **(b)** $A = 0, B = 0, C = 1$ **(c)** $A = 0, B = 0, C = 0$
(d) las respuestas (a), (b) y (c) **(e)** sólo las respuestas (a) y (b)

5. Se aplica un impulso a cada una de las entradas de una puerta NAND de dos entradas. Para $t = 0$ un impulso está a nivel ALTO y pasa a nivel BAJO en el instante $t = 1$ ms. El otro impulso está a nivel ALTO en $t = 0{,}8$ ms y pasa a nivel BAJO en $t = 3$ ms. El impulso de salida puede describirse como sigue:
(a) A nivel BAJO en $t = 0$, y pasará a nivel ALTO en $t = 3$ ms.
(b) A nivel BAJO en $t = 0{,}8$ ms, y pasará a nivel ALTO en $t = 3$ ms.
(c) A nivel BAJO en $t = 0{,}8$ ms, y pasará a nivel ALTO en $t = 1$ ms.
(d) A nivel ALTO en $t = 0{,}8$ ms, y pasará a nivel BAJO en $t = 1$ ms.

6. Se aplica un impulso a cada una de las entradas de una puerta NOR de dos entradas. Para $t = 0$ un impulso está a nivel ALTO y pasa a nivel BAJO en el instante $t = 1$ ms. El otro impulso está a nivel ALTO en $t = 0{,}8$ ms y pasa a nivel BAJO en $t = 3$ ms. El impulso de salida puede describirse como sigue:

(a) A nivel BAJO en $t = 0$, y pasará a nivel ALTO en $t = 3$ ms.
(b) A nivel BAJO en $t = 0{,}8$ ms, y pasará a nivel ALTO en $t = 3$ ms.
(c) A nivel BAJO en $t = 0{,}8$ ms, y pasará a nivel ALTO en $t = 1$ ms.
(d) A nivel ALTO en $t = 0{,}8$ ms, y pasará a nivel BAJO en $t = 1$ ms.

7. Se aplica un impulso a cada una de las entradas de una puerta OR-exclusiva. Para $t = 0$ un impulso está a nivel ALTO y pasa a nivel BAJO en el instante $t = 1$ ms. El otro impulso está a nivel ALTO en $t = 0{,}8$ ms y pasa a nivel BAJO en $t = 3$ ms. El impulso de salida estará:

(a) A nivel BAJO en $t = 0$, y pasará a nivel BAJO en $t = 3$ ms.
(b) A nivel BAJO en $t = 0$, y pasará a nivel BAJO en $t = 0{,}8$ ms.
(c) A nivel ALTO en $t = 1$ ms, y pasará a nivel ALTO en $t = 3$ ms.
(d) las respuestas (b) y (c)

8. Se aplica un impulso positivo a un inversor. El intervalo de tiempo entre el flanco anterior de la entrada y el flanco anterior de la salida es 7 ns. Este parámetro es

(a) el producto velocidad-potencia **(b)** el retardo de propagación t_{PHL}
(c) el retardo de propagación t_{PLH} **(d)** el ancho del impulso

9. La familia CMOS con la velocidad de conmutación más alta es

(a) AC **(b)** HC **(c)** ACT **(d)** ALVC

10. Si el consumo de potencia fuera el criterio principal en el diseño de un sistema digital, la familia lógica que probablemente utilizaría sería:

(a) HCT **(b)** ALVC **(c)** LV **(d)** LVC

11. Para medir el periodo de un tren de impulsos, se debe usar

(a) un multímetro digital **(b)** una sonda lógica
(c) un osciloscopio **(d)** un pulsador lógico

12. Una vez medido el periodo de un tren de impulsos, la frecuencia se calcula

(a) utilizando otro ajuste **(b)** midiendo el ciclo de trabajo
(c) como el recíproco del periodo **(d)** usando otro tipo de instrumento

■ PROBLEMAS

SECCIÓN 3.1 El inversor

1. La señal de entrada mostrada en la Figura 3.70 se aplica a un inversor. Dibujar el diagrama de tiempos de la señal de salida respecto a su entrada.

2. En la Figura 3.71 se muestra una red de inversores en cascada. Si se aplica un nivel ALTO en el punto A, determinar los niveles lógicos de los puntos B hasta F.

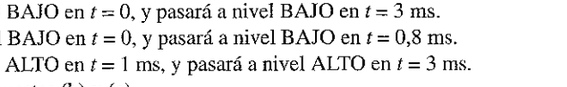


FIGURA 3.70

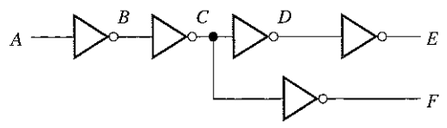


FIGURA 3.71

SECCIÓN 3.2 La puerta AND

3. Determinar la salida X para una puerta AND de dos entradas a la que se la aplican las señales de entrada mostradas en la Figura 3.72. Mostrar las relaciones de tiempo de la salida y las entradas mediante un cronograma.

4. Repetir el problema 3 para las señales de la Figura 3.73.

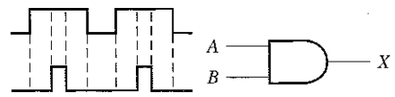


FIGURA 3.72

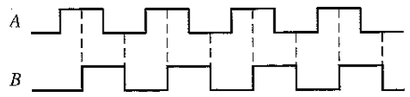


FIGURA 3.73

5. Las señales que se aplican a una puerta AND de tres entradas son las que se indican en la Figura 3.74. Determinar la señal de salida para las entradas dadas en función del tiempo, utilizando un diagrama de tiempos.

6. En la Figura 3.75 se indican las señales de entrada que se aplican a una puerta AND de cuatro entradas. Determinar la señal de salida para las entradas dadas en función del tiempo, mediante un cronograma.

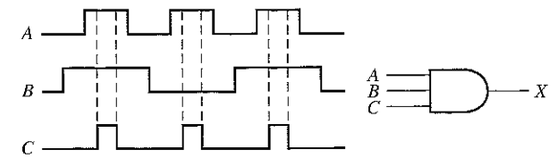


FIGURA 3.74

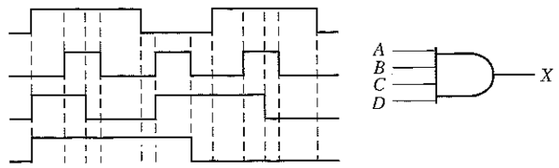


FIGURA 3.75

SECCIÓN 3.3 La puerta OR

7. Determinar la salida de una puerta OR de dos entradas cuando se aplican las señales de entrada dadas en la Figura 3.73 y dibujar el diagrama de tiempos.

8. Repetir el problema 5 para una puerta OR de 3 entradas.

9. Repetir el problema 6 para una puerta OR de 4 entradas.

10. Para las cinco señales de entrada de la Figura 3.76, determinar la salida en una puerta AND de 5 entradas y de una puerta OR de 5 entradas. Dibujar el diagrama de tiempos.

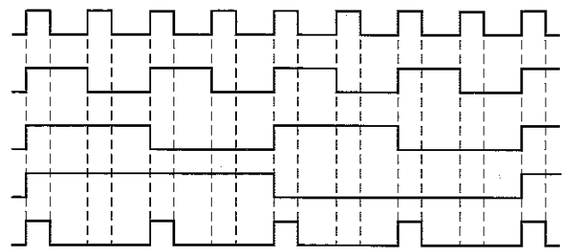


FIGURA 3.76

SECCIÓN 3.4 La puerta NAND

11. Para el conjunto de señales de entrada de la Figura 3.77, determinar la salida de la puerta mostrada y dibujar el diagrama de tiempos.

12. Determinar la salida de la puerta para las señales de entrada de la Figura 3.78 y dibujar el diagrama de tiempos.

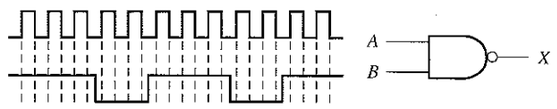


FIGURA 3.77

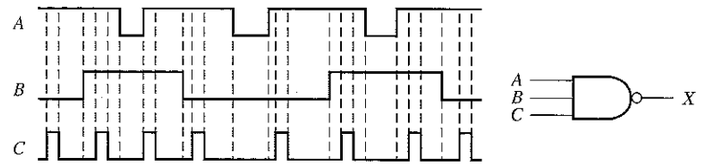


FIGURA 3.78

13. Determinar la señal de la salida correspondiente a la Figura 3.79.

14. Como se sabe, los dos símbolos lógicos representados en la Figura 3.80 representan operaciones equivalentes. La diferencia entre ellos es estrictamente de tipo funcional. Para el símbolo NAND, se requieren dos entradas a nivel ALTO para obtener una salida a nivel BAJO. Para el símbolo negativa-OR se requiere al menos una entrada a nivel BAJO para obtener una salida a nivel ALTO. Utilizando estos dos puntos de vista funcionales, demostrar que producirán la misma salida para las entradas dadas.

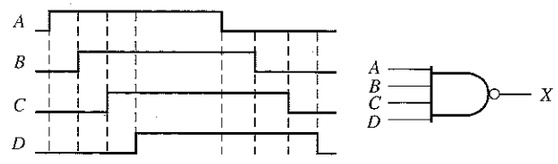


FIGURA 3.79

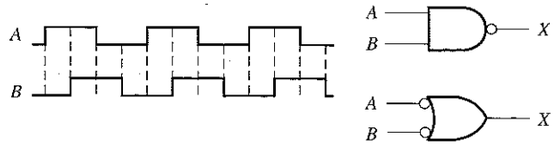


FIGURA 3.80

SECCIÓN 3.5 La puerta NOR

15. Repetir el problema 11 para una puerta NOR de 2 entradas.

16. Determinar la señal de salida para las entradas indicadas en la Figura 3.81, y dibujar el diagrama de tiempos.

17. Repetir el problema 13 para una puerta NOR de 4 entradas.

18. Los símbolos de las puertas NAND y negativa-OR representan operaciones equivalentes, pero son funcionalmente diferentes. Para la puerta NOR, se necesita al menos una de las entradas a nivel ALTO para obtener un nivel BAJO de salida. Para la puerta negativa-AND, se necesita que las dos entradas estén a nivel BAJO para obtener un nivel de salida ALTO. Utilizando estos dos puntos de vista funcionales, demostrar que ambas puertas de la Figura 3.82 generarán la misma salida para las entradas dadas.

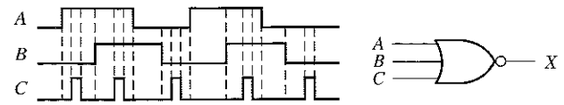


FIGURA 3.81

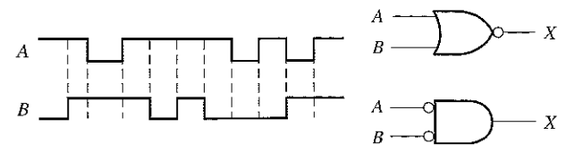


FIGURA 3.82

SECCIÓN 3.6 Puertas OR-exclusiva y NOR-exclusiva

19. ¿En qué difiere la operación lógica de la puerta OR-exclusiva de la puerta OR?

20. Repetir el problema 11 para una puerta OR-exclusiva.

21. Repetir el problema 11 para una puerta NOR-exclusiva.

22. Determinar la salida de una puerta OR-exclusiva para las entradas indicadas en la Figura 3.73, y dibujar el diagrama de tiempos.

SECCIÓN 3.7 Puertas lógicas integradas

23. En la comparación de ciertos dispositivos lógicos se ha observado que la disipación de potencia para un tipo en concreto aumenta cuando aumenta la frecuencia. ¿Se trata de un dispositivo TTL o CMOS?

24. Utilizando las hojas de características de las Figuras 3.57 y 3.58, determinar lo siguiente:

(a) La disipación de potencia del 74LS00 para la máxima tensión de alimentación y un ciclo de trabajo del 50%.

(b) La tensión de salida mínima para el nivel ALTO de un 74LS00.

(c) El retardo de propagación máximo para un 74LS00.

(d) La tensión de salida máxima para el nivel BAJO de un 74HC00A.

(e) El retardo de propagación máximo para un 74HC00A.

25. Determinar t_{PLH} y t_{PHL} para las señales de la pantalla del osciloscopio de la Figura 3.83. Las lecturas están expresadas en V/div y segundos/división para cada canal.

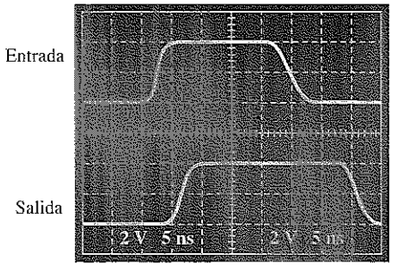


FIGURA 3.83

26. La puerta *A* tiene $t_{PLH} = t_{PHL} = 6$ ns. La puerta *B* tiene $t_{PLH} = t_{PHL} = 10$ ns. ¿Qué puerta puede trabajar a una frecuencia más alta?

27. Si una puerta lógica trabaja con una tensión de alimentación continua de + 5 V y circula una corriente media de 4 mA, ¿cuál es la potencia que disipa?

28. La variable I_{CCH} representa la corriente continua de alimentación procedente de V_{CC} cuando todas las salidas de un CI están a nivel ALTO. La variable I_{CCL} representa la corriente de alimentación continua cuando todas las salidas están a nivel BAJO. Para el CI 74LS00, determinar la disipación de potencia típica cuando las salidas de las cuatro puertas están a nivel ALTO. Consulte en el Apéndice A la hoja de características.

SECCIÓN 3.8 Localización de averías

29. Examinar las condiciones indicadas en la Figura 3.84, e identificar las puertas que fallan.

30. Determinar las puertas que fallan de la Figura 3.85 analizando los cronogramas.

31. Utilizando una sonda lógica y un pulsador, se realizan las observaciones indicadas en la Figura 3.86. Para cada observación, determinar la puerta que es más probable que falle.

32. El circuito de alarma del cinturón de seguridad de la Figura 3.16 funciona mal. Se detecta que cuando se enciende el interruptor de arranque y el cinturón está abrochado, la alarma se activa y no se apaga. ¿Cuál será el problema más probable? ¿Cómo lo localizaría?

33. Cada vez que se enciende el interruptor de arranque del circuito de la Figura 3.16, la alarma se activa durante treinta segundos, incluso cuando el cinturón está abrochado. ¿Cuál es la causa más probable de este mal funcionamiento?

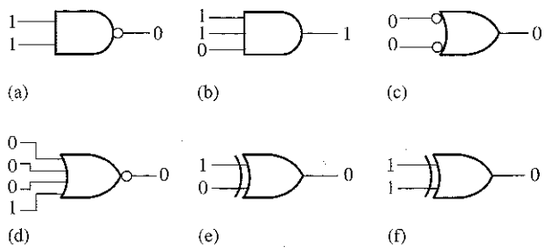


FIGURA 3.84

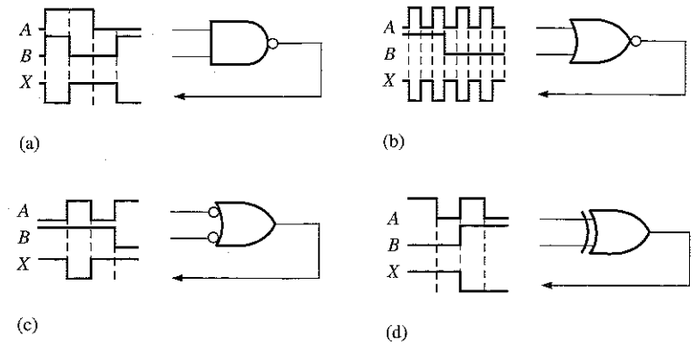


FIGURA 3.85

Aplicación a sistemas

SECCIÓN 3.9 Aplicación a los sistemas digitales

34. En el problema que se ha tratado en esta sección, realmente la lógica de inhibición del contador sólo resuelve parte del problema de temporización. Impide que el contador avance un impulso de más, haciendo que la salida del comparador pase a nivel BAJO. ¿Cuál es el problema que deja sin resolver?

35. Se supone que el sistema de control y recuento de pastillas se inicializa para 75 pastillas por bote. Dibujar el diagrama de tiempos que se vería en el analizador lógico para diez impulsos procedentes del sensor, incluyendo el impulso de cuenta final. Utilice como guía la pantalla del analizador lógico de la Práctica de sistemas 1.

36. Determinar los niveles lógicos (expresados como 1s y 0s) que aparecerían en las entradas *a* hasta *h* de la tarjeta del sistema de la Práctica de sistemas 1, para cada uno de los siguientes valores preseleccionados de pastillas por bote:

(a) 30 pastillas/bote **(b)** 75 pastillas/bote **(c)** 99 pastillas/bote

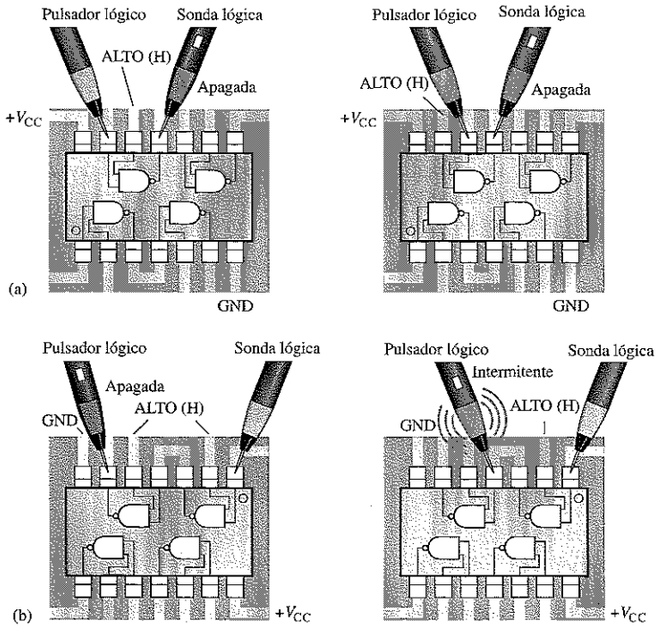


FIGURA 3.86

37. Suponer que el sistema de control y recuento de pastillas se inicializa a 50 pastillas por bote, pero sólo se introducen 18 pastillas en cada bote. Enumerar los posibles fallos del sistema que causarían este problema, en términos de las entradas a la tarjeta del sistema especificada en la Práctica de sistemas 1.

38. Con la tarjeta fuera del sistema y colocada en el banco de trabajo, sugerir un procedimiento para probar el comparador lógico.

Problemas especiales de diseño

39. Se utilizan sensores para supervisar la presión y la temperatura de una solución química almacenada en un recipiente. La circuitería de cada sensor genera un nivel de tensión ALTO cuando se excede un valor máximo especificado. Cuando se excede la presión o la temperatura, se debe activar una alarma que requiere un nivel de tensión de entrada BAJO. Diseñar un circuito para esta aplicación.

40. En un determinado proceso de fabricación automatizado, se insertan automáticamente los componentes en una tarjeta de circuito impreso. Después de activar la herramienta de inserción, la tarjeta de circuito impreso debe estar correctamente posicionada, y el componente que se va a insertar debe estar en la recámara. Estas condiciones previas se indican mediante un nivel de tensión ALTO. La herramienta de inserción requiere un nivel de tensión BAJO para activarse. Diseñar un circuito para implementar este proceso.

41. Modificar el contador de frecuencia de la Figura 3.15 para que opere con un impulso de activación (enable) que sea activo a nivel BAJO, en lugar de a nivel ALTO, durante el intervalo de 1s.

42. Suponer que la señal de activación de la Figura 3.15 es la forma de onda indicada en la Figura 3.87. Suponer que también se dispone de la señal *B*. Diseñar un circuito que genere un impulso de reset activo a nivel ALTO para el contador, sólo durante el tiempo que la señal de activación está a nivel BAJO.

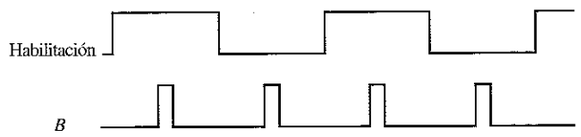


FIGURA 3.87

43. Diseñar un circuito que se colocará en el bloque rayado de la Figura 3.88, que haga que las luces delanteras de un coche se apaguen automáticamente 15 s después de que se apague el interruptor de arranque, en el caso de que el interruptor de las luces se deje activado. Suponer que se necesita un nivel BAJO para apagar las luces.

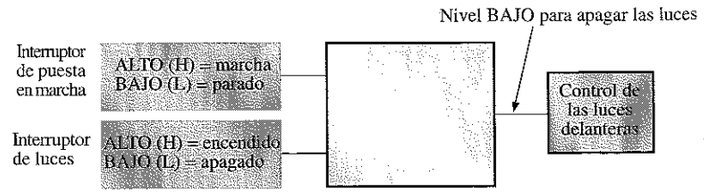


FIGURA 3.88

44. Modificar el circuito lógico de detección de intrusión de la Figura 3.24, para que se puedan proteger dos habitaciones adicionales, cada una de ellas con dos ventanas y una puerta.

45. En el sistema de control y recuento de pastillas, se ha cambiado el modelo del alimentador de pastillas y del mecanismo de la cinta transportadora. En el nuevo modelo, la válvula requiere un nivel BAJO para cerrarse y la cinta requiere un nivel BAJO para avanzar. El registro B precisa un nivel ALTO para almacenar el recuento total nuevo. Modificar la lógica del comparador para satisfacer este requisito.

Localización de averías con el Electronics Workbench

46. Abrir el archivo PR03_46.EWB y observar el funcionamiento de la puerta AND, utilizando la configuración de pruebas existente. En función de las entradas observadas y de la salida, determinar el fallo más probable que se pueda haber producido en la puerta.

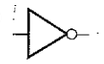
47. Abrir el archivo PR03_47.EWB y observar el funcionamiento de la puerta NAND, utilizando la configuración de pruebas existente. En función de las entradas observadas y de la salida, determinar el fallo más probable que se pueda haber producido en la puerta.

48. Abrir el archivo PR03_48.EWB y observar el funcionamiento de la puerta NOR utilizando la configuración de pruebas existente. En función de las entradas observadas y de la salida, determine el fallo más probable que se pueda haber producido en la puerta.

49. Abrir el archivo PR03_49.EWB y observar el funcionamiento de la puerta OR-exclusiva utilizando la configuración de pruebas existente. En función de las entradas observadas y de la salida, determinar el fallo más probable que se pueda haber producido en la puerta.

■ RESPUESTAS A LAS REVISIONES DE CADA SECCIÓN

SECCIÓN 3.1

1. Cuando la entrada del inversor es 1, la salida es 0.
2. **(a)** 

   (b) Hay un impulso negativo en la salida (pasa de nivel ALTO a BAJO, y vuelve a nivel ALTO).

SECCIÓN 3.2

1. La salida de una puerta AND es un nivel ALTO cuando todas las entradas están a nivel ALTO.
2. La salida de una puerta AND es un nivel BAJO cuando una o más entradas están a nivel BAJO.
3. Puerta AND de 5 entradas: $X = 1$ cuando $ABCDE = 11111$, y $X = 0$ para las restantes combinaciones de $ABCDE$.

SECCIÓN 3.3

1. La salida de una puerta OR es un nivel ALTO cuando una o más entradas están a nivel ALTO.
2. La salida de una puerta OR es un nivel BAJO cuando todas las entradas están a nivel BAJO.
3. Puerta OR de 3 entradas: $X = 0$ cuando $ABC = 000$, y $X = 1$ para las restantes combinaciones de ABC.

SECCIÓN 3.4

1. La salida de una puerta NAND es un nivel BAJO cuando todas las entradas están a nivel ALTO.
2. La salida de una puerta NAND es un nivel ALTO cuando una o más entradas están a nivel BAJO.
3. NAND: salida activa a nivel BAJO cuando todas las entradas están a nivel ALTO. Negativa-OR: salida activa a nivel ALTO cuando una o más entradas están a nivel BAJO. Ambas tienen la misma tabla de verdad.
4. $X = \overline{ABC}$

SECCIÓN 3.5

1. La salida de una puerta NOR es un nivel ALTO cuando todas las entradas están a nivel BAJO.
2. La salida de una puerta NOR es un nivel BAJO cuando una o más entradas están a nivel ALTO.
3. NOR: salida activa a nivel BAJO para una o más entradas a nivel ALTO; negativa-AND: salida activa a nivel ALTO cuando todas las entradas están a nivel BAJO. Ambas tienen la misma tabla de verdad.
4. $X = \overline{A + B + C}$

SECCIÓN 3.6

1. La salida de una puerta XOR es un nivel ALTO cuando las entradas están a niveles opuestos.
2. La salida de una puerta XNOR es un nivel ALTO cuando las entradas están al mismo nivel.
3. Aplicar los bits a las entradas de una puerta XOR. Cuando la salida está a nivel ALTO, los bits son diferentes.

SECCIÓN 3.7

1. TTL, CMOS y ECL; CMOS y TTL
2. **(a)** LS: Schottky de baja potencia **(b)** ALS: LS avanzada
   (c) F: TTL rápida **(d)** HC: CMOS de alta velocidad
   (e) AC: CMOS avanzada **(f)** HCT: HC CMOS TTL compatible
   (g) LV: CMOS de baja tensión
3. **(a)** 74LS04: inversor séxtuple **(b)** 74HC00: cuádruple NAND de 2 entradas
   (c) 74LV08: cuádruple AND de 2 entradas **(d)** 74ALS10: triple NAND de 3 entradas
   (e) 7432: cuádruple OR de 2 entradas **(f)** 74ACT11: triple AND de 3 entradas
   (g) 74AHC02: cuádruple NOR de 2 entradas
4. Menor disipación de potencia: CMOS. Mayor velocidad: ECL.
5. Seis inversores en un encapsulado; cuatro puertas NAND de dos entradas en un encapsulado.
6. $t_{PLH} = 10$ ns; $t_{PHL} = 8$ ns
7. 18 pJ
8. I_{CCL}: corriente de alimentación continua para el estado de salida BAJO; I_{CCH}: corriente de alimentación continua para el estado de salida ALTO.
9. V_{IL}: tensión de entrada para el nivel BAJO; V_{IH}: tensión de entrada para el nivel ALTO.
10. V_{OL}: tensión de salida para el nivel BAJO; V_{OH}: tensión de salida para el nivel ALTO.

SECCIÓN 3.8

1. Los fallos más comunes son los circuitos abiertos y los cortocircuitos.
2. Una entrada en circuito abierto se comporta como un nivel de entrada ALTO.
3. Las formas de onda se pueden observar con el osciloscopio pero no con la sonda lógica.
4. Una sonda es más fácil de utilizar.

SECCIÓN 3.9

1. BIT4, BIT5, BIT6 y BIT7 no cambian nunca porque el contador no excede el valor 60: 00111100.
2. $V_{izq} = 5{,}5$ V, $V_{dcha} = 0{,}4$ V
3. Las ocho entradas de la puerta NAND no pueden ponerse a nivel ALTO simultáneamente.

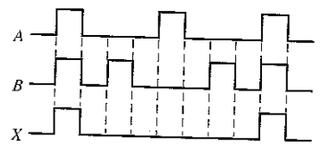
■ RESPUESTAS A LOS PROBLEMAS RELACIONADOS DE LOS EJEMPLOS

3.1 El diagrama de tiempos no varía.
3.2 Véase la Tabla 3.13.
3.3 Véase la Figura 3.89.
3.4 La forma de onda de salida es igual que la entrada *A*.

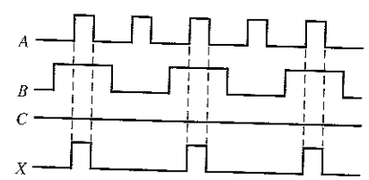
TABLA 3.13

Entradas *ABCD*	Salida *X*	Entradas *ABCD*	Salida *X*
0000	0	1000	0
0001	0	1001	0
0010	0	1010	0
0011	0	1011	0
0100	0	1100	0
0101	0	1101	0
0110	0	1110	0
0111	0	1111	1

FIGURA 3.89



3.5 Véase la Figura 3.90
3.6 Véase la Figura 3.91
3.7 Véase la Figura 3.92
3.8 Véase la Figura 3.93
3.9 Véase la Figura 3.94
3.10 Véase la Figura 3.95



C = ALTO (H)

FIGURA 3.90

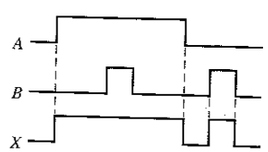


FIGURA 3.91

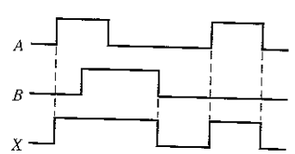
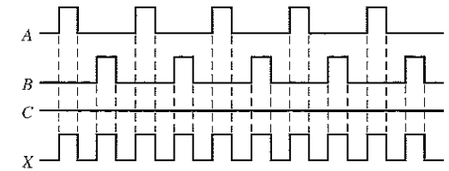


FIGURA 3.92



C = BAJO (L)

FIGURA 3.93

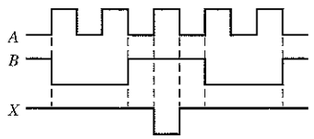


FIGURA 3.94

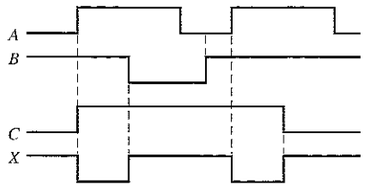


FIGURA 3.95

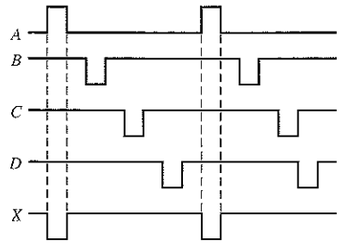


FIGURA 3.96

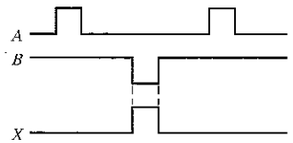


FIGURA 3.97

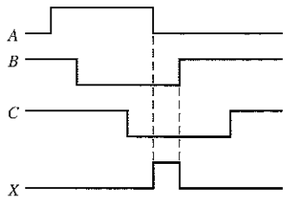


FIGURA 3.98

3.11 Utilizar una puerta NAND de 3 entradas.

3.12 Utilizar una puerta NAND de 4 entradas que funcione como una puerta OR-Negativa.

3.13 Véase la Figura 3.96.

3.14 Véase la Figura 3.97.

3.15 Véase la Figura 3.98.

3.16 Utilizar una puerta NOR de 2 entradas.

3.17 Utilizar una puerta NAND de 3 entradas.

3.18 La salida siempre está a nivel BAJO. El cronograma es una línea recta.

3.19 La puerta OR-exclusiva no detectará simultáneamente fallos si ambos circuitos generan la misma salida.

3.20 Las salidas no se ven afectadas.

3.21 La puerta con t_{PLH} y t_{PHL} igual a 4 ns puede funcionar a la frecuencia más alta.

3.22 10 mW

3.23 La salida de la puerta o el pin 13 de entrada están internamente en circuito abierto.

3.24 El display mostrará una lectura errónea porque el contador continúa hasta que se pone a cero.

3.25 El impulso de habilitación es demasiado corto o el contador se ha puesto a cero demasiado pronto.

■ RESPUESTAS AL AUTOTEST

1. (d) **2.** (d) **3.** (a) **4.** (e) **5.** (c) **6.** (a)
7. (d) **8.** (b) **9.** (d) **10.** (d) **11.** (c) **12.** (c)

4

ÁLGEBRA DE BOOLE Y SIMPLIFICACIÓN LÓGICA

4.1 Operaciones y expresiones booleanas
4.2 Leyes y reglas del Álgebra de Boole
4.3 Teoremas de DeMorgan
4.4 Análisis booleano de los circuitos lógicos
4.5 Simplificación mediante el Álgebra de Boole
4.6 Formas estándar de las expresiones booleanas
4.7 Expresiones booleanas y tablas de verdad
4.8 Mapas de Karnaugh
4.9 Minimización de una suma de productos mediante el mapa de Karnaugh
4.10 Minimización de un producto de sumas mediante el mapa de Karnaugh
4.11 Mapa de Karnaugh de cinco variables
4.12 Aplicación a los sistemas digitales

Simulación por computadora de circuitos digitales:
El tutorial de Electronics Workbench (EWB) se encuentra en la dirección http://www.prenhall.com/floyd

■ OBJETIVOS DEL CAPÍTULO

- ❑ Aplicar las leyes y reglas básicas del álgebra de Boole.
- ❑ Aplicar los teoremas de DeMorgan a las expresiones booleanas.
- ❑ Describir redes de puertas mediante expresiones booleanas.
- ❑ Evaluar las expresiones booleanas.
- ❑ Simplificar expresiones mediante las leyes y reglas del álgebra booleana.
- ❑ Convertir cualquier expresión booleana en una suma de productos.
- ❑ Convertir cualquier expresión booleana en un producto de sumas.
- ❑ Utilizar el mapa de Karnaugh para simplificar expresiones booleanas.
- ❑ Utilizar el mapa de Karnaugh para simplificar tablas de verdad.
- ❑ Utilizar condiciones no definidas para simplificar funciones lógicas.
- ❑ Aplicar el álgebra de Boole y los mapas de Karnaugh a los sistemas digitales.

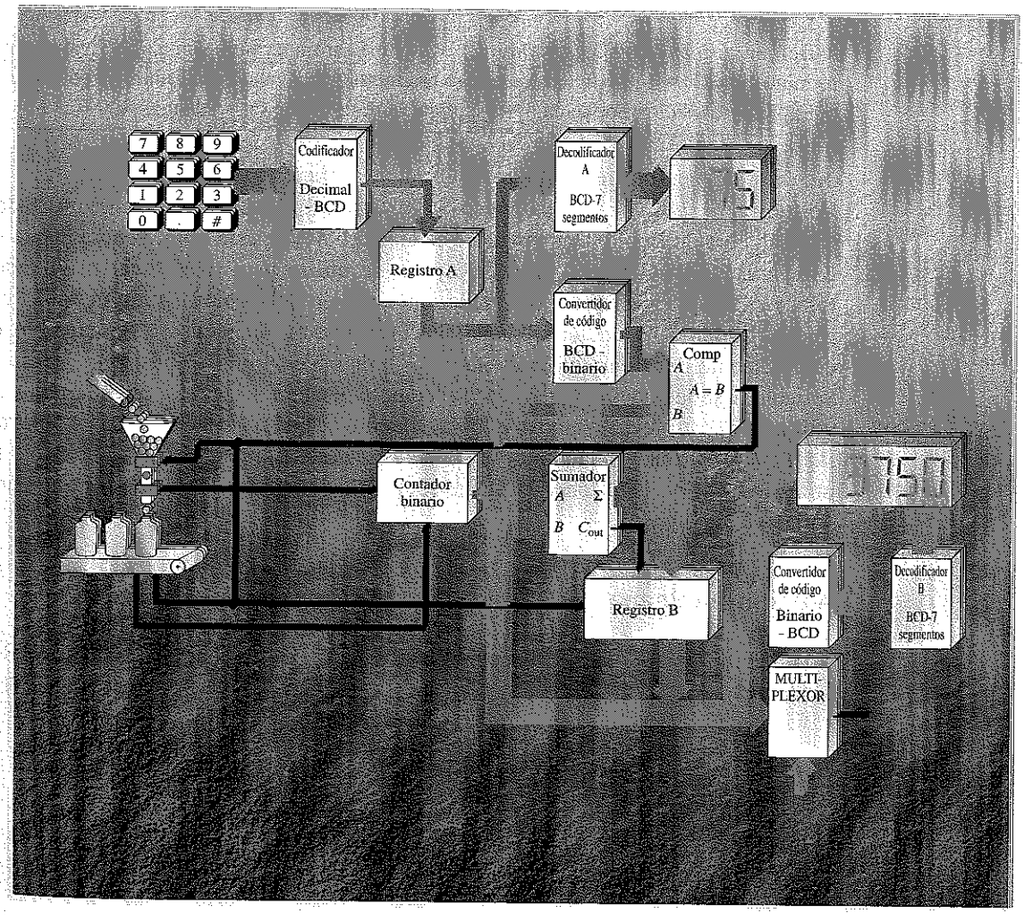
■ INTRODUCCIÓN

En 1854, George Boole publicó una obra titulada *Investigación de las leyes del pensamiento, sobre las que se basan las teorías matemáticas de la lógica y la probabilidad*. En esta publicación se formuló la idea de un "álgebra de las operaciones lógicas", que se conoce hoy en día como álgebra de Boole. El álgebra de Boole es una forma muy adecuada para expresar y analizar las operaciones de los circuitos lógicos. Claude Shannon fue el primero en aplicar la obra de Boole al análisis y diseño de circuitos. En 1938, Shannon escribió su tesis doctoral en el MIT (Massachussets Institute of Technology) titulada *Análisis simbólico de los circuitos de conmutación y relés*.

Este capítulo se ocupa de las leyes, reglas y teoremas del álgebra booleana y sus aplicaciones a los circuitos digitales. Se aprenderá a definir un circuito mediante una expresión booleana y a determinar su funcionamiento. También se tratará la simplificación de los circuitos lógicos utilizando el álgebra booleana y los mapas de Karnaugh.

■ APLICACIÓN A LOS SISTEMAS DIGITALES

Esta aplicación a los sistemas digitales ilustra los conceptos que serán explicados a lo largo del capítulo. El funcionamiento del display de 7 segmentos del sistema de control y recuento de pastillas es un buen método para ilustrar la aplicación del álgebra de Boole y de los mapas de Karnaugh, de modo que se obtenga la más sencilla implementación en el diseño de circuitos lógicos. Por tanto, en la Sección 4.12, nos centraremos en la lógica del convertidor BCD-7 segmentos que gobierna los dos displays del sistema indicados a continuación.



4.1 ■ OPERACIONES Y EXPRESIONES BOOLEANAS

NOTAS INFORMÁTICAS En un microprocesador, la unidad aritmético lógica (ALU) realiza las operaciones aritméticas y lógicas booleanas sobre los datos digitales mediante instrucciones de programa. Las operaciones lógicas son equivalentes a las operaciones de las puertas básicas con las que ya estamos familiarizados, aunque se trabaja con ocho bits como mínimo a la vez. Ejemplos de instrucciones lógicas booleanas son AND, OR, NOT y XOR, que se denominan *mnemónicos*. Un programa en lenguaje ensamblador utiliza estos mnemónicos para especificar una operación. Y otro programa denominado *ensamblador* traduce los mnemónicos a un código binario que puede entender el microprocesador.

El álgebra de Boole son las matemáticas de los sistemas digitales. Es indispensable tener unos conocimientos básicos del álgebra booleana para estudiar y analizar los circuitos lógicos. En el capítulo anterior, se han introducido las operaciones y expresiones booleanas para las puertas NOT, AND, OR, NAND y NOR. En esta sección se revisa este material y se facilitan definiciones e información adicional.

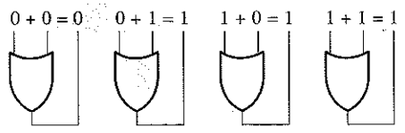
Al finalizar esta sección, el lector deberá ser capaz de:

■ Definir *variable*. ■ Definir *literal*. ■ Identificar un término suma. ■ Evaluar un término suma. ■ Identificar un término producto. ■ Evaluar un término producto. ■ Explicar la adición booleana. ■ Explicar la multiplicación booleana.

Los términos *variable, complemento* y *literal* son términos ampliamente utilizados en el álgebra de Boole. Una **variable** es un símbolo (en este libro, generalmente se usará una letra mayúscula en cursiva) que se utiliza para representar magnitudes lógicas. Cualquier variable puede tener un valor 1 o 0. El **complemento** es el inverso de la variable y se indica mediante una barra encima de la misma. Por ejemplo, el complemento de la variable A es $\overline{A}$. Si $A=1$, entonces $\overline{A}=0$. El complemento de la variable A se lee como "A negada" o "complementada". Algunas veces también se utiliza un apóstrofe para indicar el complemento de una variable en lugar de la barra. Por ejemplo, el complemento de B puede escribirse como B'. En este libro, sólo se utilizará la barra. Un **literal** se define como una variable o el complemento de una variable.

Adición booleana

Como vimos en el Capítulo 3, la **suma booleana** es equivalente a la operación OR y sus reglas son:



En el álgebra de Boole, un **término suma** es una suma de literales. En los circuitos lógicos, un término suma se produce mediante la operación OR, sin que exista ninguna operación AND en la expresión. Algunos ejemplos son: $A + B$, $A + \overline{B}$, $A + B + \overline{C}$ y $\overline{A} + B + C + \overline{D}$.

Un término suma es igual a 1 cuando uno o más de los literales es un 1. Un término suma es igual a 0 si y sólo si cada uno de los literales es 0.

EJEMPLO 4.1 Determinar los valores de A, B, C y D que hacen que el término $A+\overline{B}+C+\overline{D}$ sea igual a 0.

Solución

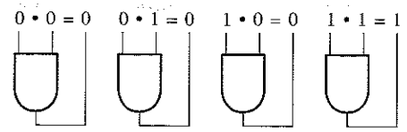
Para que la suma sea 0, cada uno de los literales tiene que ser 0. Por consiguiente, $A = \mathbf{0}$, $B = \mathbf{1}$ (para que $\overline{B} = 0$), $C = \mathbf{0}$ y $D = \mathbf{1}$ (para que $\overline{D} = 0$).

$$A + \overline{B} + C + \overline{D} = 0 + \overline{1} + 0 + \overline{1} = 0 + 0 + 0 + 0 = 0$$

Problema relacionado. Determinar los valores de A y B que hacen que el término $\overline{A} + B$ sea 0.

Multiplicación booleana

En el Capítulo 3 vimos también que la **multiplicación booleana** es equivalente a la operación AND y sus reglas básicas son:



En el álgebra de Boole, un **término producto** es el producto de literales. En los circuitos lógicos, un término producto se produce mediante la operación AND, sin que aparezca ninguna operación OR en la expresión. Algunos ejemplos de términos producto son: AB, $A\overline{B}$, ABC y $A\overline{B}C\overline{D}$.

Un término producto es igual a 1 si y sólo si cada uno de los literales es 1. Un término producto es igual a 0 si uno o más de sus literales es 0.

EJEMPLO 4.2 Determinar los valores de A, B, C y D que hacen que el producto $A\overline{B}C\overline{D}$ sea igual a 1.

Solución

Para que este producto sea 1, cada uno de los literales del término tiene que ser 1. Por tanto, $A = \mathbf{1}$, $B = \mathbf{0}$ (para que $\overline{B} = 1$), $C = \mathbf{1}$ y $D = \mathbf{0}$ (para que $\overline{D} = 1$).

$$A\overline{B}C\overline{D} = 1 \cdot \overline{0} \cdot 1 \cdot \overline{0} = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

Ejercicio relacionado. Determinar los valores de A y B que hacen que el término producto de $\overline{A}\,\overline{B}$ sea igual a 1.

REVISIÓN DE LA SECCIÓN 4.1

1. Si $A = 0$, ¿cuánto vale $\overline{A}$?
2. Determinar los valores de A, B y C que hacen que el término $\overline{A}+\overline{B}+C$ sea 0.
3. Determinar los valores de A, B y C que hacen que el término $A\overline{B}C$ sea igual a 1.

4.2 ■ LEYES Y REGLAS DEL ÁLGEBRA DE BOOLE

Al igual que en otras áreas de las matemáticas, existe en el álgebra de Boole una serie de reglas y leyes bien determinadas que tienen que seguirse para aplicarla correctamente. Las más importantes son las que se presentan en esta sección.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Aplicar las leyes conmutativas de la adición y multiplicación. ■ Aplicar las leyes asociativas de la adición y multiplicación. ■ Aplicar la ley distributiva. ■ Aplicar las doce reglas básicas del álgebra de Boole.

Leyes del álgebra de Boole

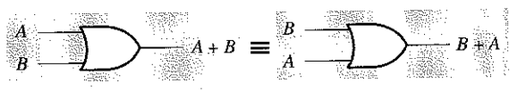
Las leyes básicas del álgebra de Boole (las **leyes conmutativas** de la suma y la multiplicación, y las **leyes asociativas** de la suma y la multiplicación y la **ley distributiva**) son las mismas que las del álgebra ordinaria. Cada una de las leyes se ilustra con dos o tres variables, pero el número de variables no está limitado a esta cantidad.

Leyes conmutativas. La *ley conmutativa de la suma* para dos variables se escribe como sigue:

$$A + B = B + A \tag{4.1}$$

Esta ley establece que el orden en que se aplica a las variables la operación OR es indiferente. Recuerde que cuando se aplica a los circuitos lógicos la terminología del álgebra de Boole, la adición y la operación OR son una misma cosa. La Figura 4.1 nos muestra la ley conmutativa aplicada a una puerta OR, en la que se puede ver que es indistinto a qué entrada asignemos cada una de las variables. (El símbolo ≡ significa "equivalente a".)

FIGURA 4.1
Aplicación de la ley conmutativa de la adición.

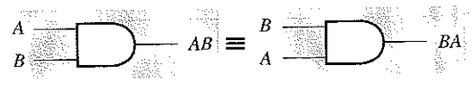


La ley conmutativa de la multiplicación para dos variables es:

$$AB = BA \tag{4.2}$$

Esta ley establece que el orden en que se aplica a las variables la operación AND es indiferente. La Figura 4.2 muestra esta ley cuando se aplica a una puerta AND.

FIGURA 4.2
Aplicación de la ley conmutativa de la multiplicación.

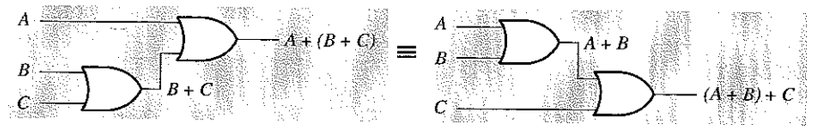


Leyes asociativas. La ley asociativa de la adición para tres variables se escribe, en forma algebraica, de la siguiente manera:

$$A + (B + C) = (A + B) + C \tag{4.3}$$

Esta ley establece que al aplicar la operación OR a más de dos variables, el resultado es el mismo independientemente de la forma en que se agrupen las variables. En la Figura 4.3 podemos ver esta ley aplicada a las puertas OR.

FIGURA 4.3
Aplicación de la ley asociativa de la adición.

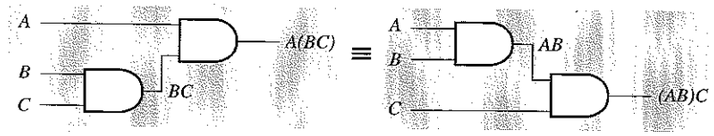


La ley asociativa de la multiplicación para tres variables se escribe de la siguiente manera:

$$A(BC) = (AB)C \tag{4.4}$$

Esta ley establece que al aplicar la operación AND a más de dos variables, el resultado es el mismo independientemente de la forma en que se agrupen las variables. En la Figura 4.4 podemos ver esta ley aplicada a las puertas AND.

FIGURA 4.4
Aplicación de la ley asociativa de la multiplicación.



Ley distributiva. La ley distributiva para tres variables se escribe de la siguiente manera:

$$A(B + C) = AB + AC \tag{4.5}$$

Esta ley establece que aplicar la operación OR a dos o más variables y luego aplicar la operación AND al resultado de esta operación y a otra variable aislada, es equivalente a aplicar la operación AND a la variable aislada con cada uno de los sumandos y luego aplicar la operación OR a los productos resultantes. La ley distributiva también expresa el proceso de sacar factor común, en el que la variable común A se saca como factor de los productos parciales, como, por ejemplo, en $AB + AC = A(B + C)$. La Figura 4.5 ilustra la ley distributiva mediante su implementación de puertas.

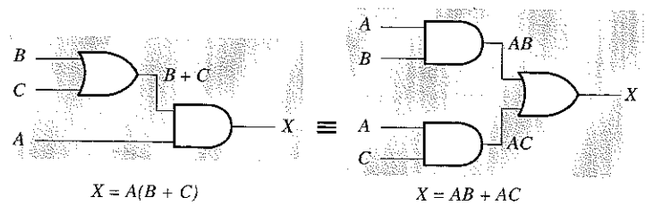


FIGURA 4.5
Aplicación de la ley distributiva.

Reglas del álgebra booleana

La Tabla 4.1 enumera las doce reglas básicas, muy útiles, para la manipulación y simplificación de **expresiones booleanas**. Las nueve primeras reglas las veremos en términos de su aplicación a las puertas lógicas. Las reglas 10 a 12 se obtendrán a partir de las reglas más sencillas y de las leyes anteriormente explicadas.

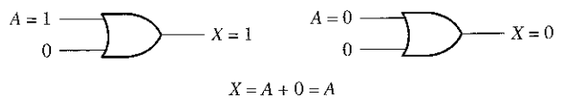
TABLA 4.1
Reglas básicas del álgebra de Boole.

1. $A + 0 = A$	**7.** $A \cdot A = A$
2. $A + 1 = 1$	**8.** $A \cdot \overline{A} = 0$
3. $A \cdot 0 = 0$	**9.** $\overline{\overline{A}} = A$
4. $A \cdot 1 = A$	**10.** $A + AB = A$
5. $A + A = A$	**11.** $A + \overline{A}B = A + B$
6. $A + \overline{A} = 1$	**12.** $(A + B)(A + C) = A + BC$

A, B, o *C* pueden representar una única variable o una combinación de variables.

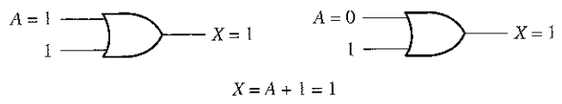
1. $A + 0 = A$ Si aplicamos la operación OR a una variable cualquiera y a 0, el resultado es siempre igual a la variable. Si A es 1, la salida es igual a 1 y, por tanto, igual a A. Si A es 0, la salida es 0 e igualmente idéntica a A. Esta ley se ilustra en la Figura 4.6, en la que la entrada inferior está siempre a 0.

FIGURA 4.6
***Regla 1:* $A + 0 = A$.**



2. $A + 1 = 1$ Si se aplica la operación OR a una variable y a 1, el resultado es siempre igual a 1. Un 1 en una entrada de una puerta OR produce siempre un 1 en la salida, independientemente del valor de la otra entrada. Esta regla se ilustra en la Figura 4.7, en la que la entrada inferior está siempre a 1.

FIGURA 4.7
***Regla 2:* $A + 1 = 1$.**



3. $A \cdot 0 = 0$ Si se aplica la operación AND a una variable y a 0, el resultado es siempre igual a 0. Siempre que una de las entradas de una puerta AND sea 0, la salida es siempre 0, independientemente del valor de la otra entrada. Esta regla se ilustra en la Figura 4.8, en la que la entrada inferior está siempre a 0.

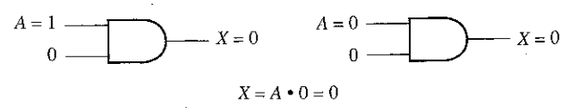
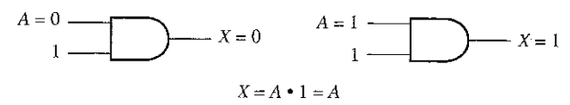


FIGURA 4.8
***Regla 3:* $A \cdot 0 = 0$.**

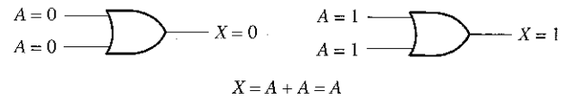
4. $A \cdot 1 = A$ Si se aplica la operación AND a 1 y a otra variable, el resultado es siempre igual a la variable. Si la variable A es 0, la salida de la puerta AND será siempre 0, mientras que si A es 1, la salida será 1, dado que las dos entradas son 1. Esta regla se ilustra en la Figura 4.9, en la que la entrada inferior está siempre a 1.

FIGURA 4.9
***Regla 4:* $A \cdot 1 = A$.**



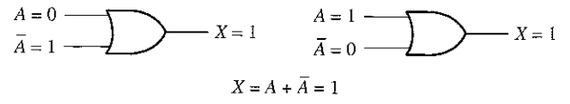
5. $A + A = A$ Si se aplica la operación OR a una variable consigo misma, el resultado es siempre igual a la variable. Si A es 0, entonces $0 + 0 = 0$, mientras que si A es 1, $1 + 1 = 1$. Esto se muestra en la Figura 4.10, en la que se aplica la misma variable a ambas entradas.

FIGURA 4.10
***Regla 5:* $A + A = A$.**



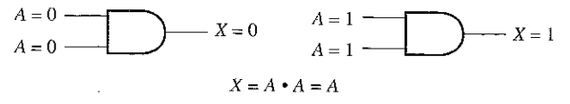
6. $A + \overline{A} = 1$ Si se aplica la operación OR a una variable y a su complemento, el resultado es siempre igual a 1. Si A es 0, entonces $0 + \overline{0} = 0 + 1 = 1$. Si A es 1, entonces $1 + \overline{1} = 1 + 0 = 1$. En la Figura 4.11 podemos ver una puerta OR en la que sus entradas son una variable y su complemento.

FIGURA 4.11
***Regla 6:* $A + \overline{A} = 1$.**



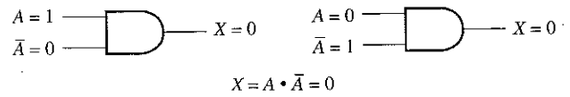
7. $A \cdot A = A$ Si se aplica la operación AND a una variable consigo misma, el resultado siempre es igual a la variable. Si $A = 0$, entonces $0 \cdot 0 = 0$, y si $A = 1$, entonces $1 \cdot 1 = 1$. Esta regla se ilustra en la Figura 4.12.

FIGURA 4.12
***Regla 7:* $A \cdot A = A$.**



8. $A \cdot \overline{A} = 0$ Si se aplica la operación AND a una variable y a su complemento, el resultado es siempre igual a 0. Esta regla se basa en que siempre A o $\overline{A}$ será 0, y además en que cuando se aplica un 0 a una de las entradas de una puerta AND la salida siempre es 0. En la Figura 4.13 se muestra esta regla.

FIGURA 4.13
Regla 8: $A \cdot \overline{A} = 0$.



9. $\overline{\overline{A}} = A$ El complemento del complemento de una variable es siempre la propia variable. El complemento de la variable A es $\overline{A}$ y el complemento de $\overline{A}$ será de nuevo A, que es la variable original. Esta regla se muestra en la Figura 4.14, mediante el uso de dos inversores.

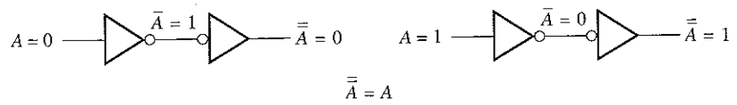


FIGURA 4.14
Regla 9: $\overline{\overline{A}} = A$.

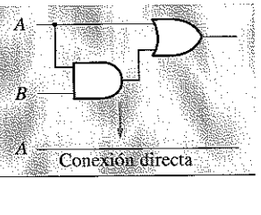
10. $A + AB = A$ Esta regla se puede obtener aplicando la ley distributiva, reglas 2 y 4, de la manera siguiente:

$$\begin{aligned} A + AB &= A\,(1 + B) && \text{Sacar factor común } A \text{ (ley distributiva)} \\ &= A \cdot 1 && \text{Regla 2: } (1 + B) = 1 \\ &= A && \text{Regla 4: } A \cdot 1 = A \end{aligned}$$

En la Tabla 4.2 podemos ver la tabla de verdad y la simplificación del circuito lógico resultante.

TABLA 4.2

A	B	AB	A + AB
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1



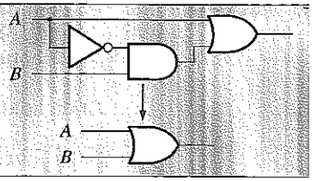
11. $A + \overline{A}B = A + B$ Esta regla se puede demostrar de la manera siguiente:

$$\begin{aligned} A + \overline{A}B &= (A + AB) + \overline{A}B && \text{Regla 10: } A = A + AB \\ &= \mathrm{A} + (\mathrm{A} + \overline{\mathrm{A}})\,\mathrm{B} && \text{Sacar factor común} \\ &= \mathrm{A} + 1 \cdot \mathrm{B} && \text{Regla 6: } \mathrm{A} + \overline{\mathrm{A}} = 1 \\ &= A + \mathrm{B} && \text{Regla 4: } \mathrm{A} \cdot 1 = \mathrm{A} \end{aligned}$$

En la Tabla 4.3 podemos ver la tabla de verdad y la simplificación del circuito lógico resultante.

TABLA 4.3

A	B	$\overline{A}B$	A + AB	A + B
0	0	0	0	0
0	1	1	1	1
1	0	0	1	1
1	1	0	1	1



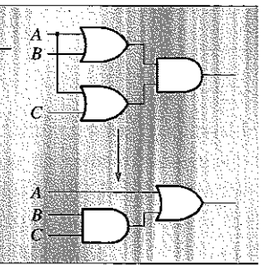
12. $(A + B)(A + C) = A + BC$ Esta regla se puede demostrar de la manera siguiente:

$$\begin{aligned} (A + B)(A + C) &= AA + AC + AB + BC && \text{Ley distributiva} \\ &= A + AC + AB + BC && \text{Regla 7: } AA = A \\ &= A + BC && \text{Regla 10: } A + AB = A \\ & && \text{(aplicada 2 veces)} \end{aligned}$$

En la Tabla 4.4 podemos ver la tabla de verdad y la simplificación del circuito lógico resultante.

TABLA 4.4

A	B	C	A + B	A + C	(A + B)(A + C)	BC	A + BC
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1



REVISIÓN DE LA SECCIÓN 4.2

1. Aplicar la ley asociativa de la adición a la expresión $A + (B + C + D)$.
2. Aplicar la ley distributiva a la expresión $A(B + C + D)$.

4.3 ■ TEOREMAS DE DEMORGAN

DeMorgan, matemático que conoció a Boole, propuso dos teoremas que constituyen una parte muy importante del álgebra de Boole. En términos prácticos, los teoremas de DeMorgan nos demuestran la equivalencia entre las puertas NAND y negativa-OR, y las puertas NOR y negativa-AND, que se han tratado en el Capítulo 3.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Enunciar los teoremas de DeMorgan. ■ Relacionar los teoremas de DeMorgan con la equivalencia entre las puertas NAND y negativa-OR, y entre las puertas NOR y negativa-AND. ■ Aplicar los teoremas de DeMorgan para simplificar las expresiones booleanas.

El primer teorema de DeMorgan se enuncia de la siguiente forma:

El complemento de un producto de variables es igual a la suma de los complementos de las variables.

O dicho de otra manera,

El complemento de dos o más variables a las que se aplica la operación AND es equivalente a aplicar la operación OR a los complementos de cada variable.

La fórmula para expresar este teorema para dos variables es:

$$\overline{XY} = \overline{X} + \overline{Y} \tag{4.6}$$

El segundo teorema de DeMorgan se enuncia así:

El complemento de una suma de variables es igual al producto de los complementos de las variables.

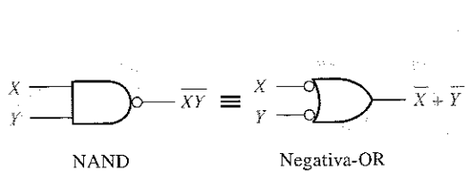
O dicho de otra manera,

El complemento de dos o más variables a las que se aplica la operación OR es equivalente a aplicar la operación AND a los complementos de cada variable.

La fórmula para expresar este teorema es:

$$\overline{X+Y} = \overline{X}\,\overline{Y} \tag{4.7}$$

Las puertas equivalentes y tablas de verdad correspondientes a las ecuaciones 4.6 y 4.7 se muestran en la Figura 4.15.



Entradas		Salida	
X	Y	$\overline{XY}$	$\overline{X}+\overline{Y}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

Entradas		Salida	
X	Y	$\overline{X+Y}$	$\overline{X}\,\overline{Y}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

FIGURA 4.15
Equivalencias de las puertas lógicas y tablas de verdad que ilustran los teoremas de DeMorgan. Nótese la igualdad entre las dos columnas de salida de cada tabla. Esto indica que las puertas equivalentes realizan la misma función.

Como se ha comentado, los teoremas de DeMorgan se aplican también a expresiones en las que existen más de dos variables. Los siguientes ejemplos ilustran la aplicación de los teoremas de DeMorgan a expresiones de 3 y 4 variables.

EJEMPLO 4.3 Aplicar los teoremas de DeMorgan a las expresiones $\overline{XYZ}$ y $\overline{X+Y+Z}$.

Solución

$$\overline{XYZ} = \overline{X} + \overline{Y} + \overline{Z}$$
$$\overline{X+Y+Z} = \overline{X}\,\overline{Y}\,\overline{Z}$$

Problema relacionado. Aplicar los teoremas de DeMorgan a la expresión $\overline{\overline{X} + \overline{Y} + \overline{Z}}$

EJEMPLO 4.4 Aplicar los teoremas de DeMorgan a las expresiones $\overline{WXYZ}$ y $\overline{W+X+Y+Z}$.

Solución

$$\overline{WXYZ} = \overline{W} + \overline{X} + \overline{Y} + \overline{Z}$$
$$\overline{W+X+Y+Z} = \overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z}$$

Problema relacionado. Aplicar los teoremas de DeMorgan a la expresión $\overline{\overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z}}$

Como se ha establecido en las Ecuaciones 4.6 y 4.7 que enuncian los teoremas de DeMorgan, cada variable puede representar una combinación de otras variables. Por ejemplo, X puede ser igual al término $AB+C$, e Y puede ser igual a $A+BC$. De esta forma, si aplicamos el teorema de DeMorgan para dos variables expresado según $\overline{XY} = \overline{X} + \overline{Y}$ a la expresión $\overline{(AB + C)(A + BC)}$ obtenemos el siguiente resultado:

$$\overline{(AB + C)(A + BC)} = \overline{(AB + C)} + \overline{(A + BC)}$$

En el resultado anterior se observa que hay dos términos, $\overline{AB + C}$ y $\overline{A + BC}$, a los que podemos aplicar otra vez el teorema de DeMorgan $\overline{X} + \overline{Y} = \overline{XY}$, obteniendo:

$$\overline{(AB + C)} + \overline{(A + BC)} = (\overline{AB})\overline{C} + \overline{A}(\overline{BC})$$

De esta manera obtenemos otros dos términos en la expresión a los que de nuevo podemos aplicar el teorema de DeMorgan. Estos términos son $\overline{AB}$ y $\overline{BC}$. Una última aplicación del teorema de DeMorgan nos proporciona el siguiente resultado:

$$(\overline{AB})\overline{C} + \overline{A}(\overline{BC}) = (\overline{A} + \overline{B})\overline{C} + \overline{A}(\overline{B} + \overline{C})$$

Los teoremas de DeMorgan ya no pueden seguir aplicándose, aunque este resultado puede ser simplificado aún más mediante el uso de las reglas y leyes de Boole.

Aplicación de los teoremas de DeMorgan

El siguiente procedimiento ilustra la aplicación de los teoremas de DeMorgan y del álgebra de Boole a una determinada expresión (los corchetes son equivalentes a los paréntesis):

$$\overline{\overline{A + B\overline{C}} + D(\overline{E + \overline{F}})}$$

Paso 1. Identificar los términos a los que se puede aplicar los teoremas de DeMorgan y considerar cada término como una única variable. Definimos $\overline{A + B\overline{C}} = X$ y $D(\overline{E + \overline{F}}) = Y$

Paso 2. Dado que $\overline{X + Y} = \overline{X}\,\overline{Y}$,

$$\overline{[A + B\overline{C}] + [D(\overline{E + \overline{F}})\,]} = \overline{[A + B\overline{C}]}\,\overline{[D(\overline{E + \overline{F}})]}$$

Paso 3. Utilizar la regla 9 ($\overline{\overline{A}} = A$) para eliminar la barra doble sobre el término de la izquierda (esta parte no tiene que ver con los teoremas de DeMorgan):

$$[A + B\overline{C}]\,\overline{[D(\overline{E + \overline{F}})]} = [A + B\overline{C}]\,\overline{[D(\overline{E + \overline{F}})]}$$

Paso 4. En el término de la derecha definimos $Z = \overline{E + \overline{F}}$

$$[A + B\overline{C}]\,\overline{[D(\overline{E + \overline{F}})]} = [A + B\overline{C}]\,[\overline{DZ}]$$

Paso 5. Como $\overline{DZ} = \overline{D} + \overline{Z}$,

$$[A + B\overline{C}]\,\overline{[D(\overline{E + \overline{F}})]} = (A + B\overline{C})(\overline{D} + \overline{\overline{E + \overline{F}}})$$

Paso 6. Utilizando la regla 9 ($\overline{\overline{A}} = A$) para eliminar la barra doble del término $E + \overline{F}$:

$$(A + B\overline{C})(\overline{D} + \overline{\overline{E + \overline{F}}}) = (A + B\overline{C})(\overline{D} + E + \overline{F})$$

Los siguientes tres ejemplos ilustrarán detalladamente cómo emplear los teoremas de DeMorgan.

EJEMPLO 4.5 Aplicar los teoremas de DeMorgan a las expresiones:

(a) $\overline{(A+B+C)D}$ **(b)** $\overline{ABC+DEF}$ **(c)** $\overline{A\overline{B} + \overline{C}D + EF}$

Solución

(a) Sean $A + B + C = X$ y $D = Y$. La expresión $\overline{(A+B+C)D}$ está escrita en la forma $\overline{XY} = \overline{X} + \overline{Y}$ y puede expresarse como

$$\overline{(A+B+C)D} = \overline{A+B+C} + \overline{D}$$

A continuación, aplicamos el teorema de DeMorgan al término $\overline{A+B+C}$

$$\overline{A+B+C} + \overline{D} = \overline{A}\,\overline{B}\,\overline{C} + \overline{D}$$

(b) Sean $ABC = X$ y $DEF = Y$. La expresión $\overline{ABC+DEF}$ está escrita en la forma $\overline{X+Y} = \overline{X}\,\overline{Y}$, y puede expresarse como:

$$\overline{ABC + DEF} = (\overline{ABC})(\overline{DEF})$$

A continuación aplicamos el teorema de DeMorgan a cada uno de los términos $\overline{ABC}$ y $\overline{DEF}$:

$$(\overline{ABC})(\overline{DEF}) = (\overline{A} + \overline{B} + \overline{C})(\overline{D} + \overline{E} + \overline{F})$$

(c) Sean $A\overline{B} = X$, $\overline{C}D = Y$, y $EF = Z$. La expresión $\overline{A\overline{B} + \overline{C}D + EF}$ está escrita en la forma $\overline{X + Y + Z} = \overline{X}\,\overline{Y}\,\overline{Z}$, y puede expresarse como:

$$\overline{A\overline{B} + \overline{C}D + EF} = (\overline{A\overline{B}})(\overline{\overline{C}D})(\overline{EF})$$

A continuación aplicamos el teorema de DeMorgan a cada uno de los términos $\overline{A\overline{B}}$, $\overline{\overline{C}D}$, y $\overline{EF}$:

$$(\overline{A\overline{B}})(\overline{\overline{C}D})(\overline{EF}) = (\overline{A} + B)(C + \overline{D})(\overline{E} + \overline{F})$$

Problema relacionado. Aplicar los teoremas de DeMorgan a la expresión $\overline{AB\overline{C} + D + E}$

EJEMPLO 4.6 Aplicar los teoremas de DeMorgan a las expresiones:

(a) $\overline{(\overline{A + B}) + \overline{C}}$ **(b)** $\overline{(\overline{A} + B) + CD}$ **(c)** $\overline{(A + B)\overline{C}\,\overline{D} + E + \overline{F}}$

Solución

(a) $\overline{(\overline{A + B}) + \overline{C}} = \overline{\overline{(A + B)}\,\overline{C}} = (A + B)C$

(b) $\overline{(\overline{A} + B) + CD} = \overline{(\overline{A} + B)CD} = \overline{(\overline{A}\,\overline{B})}(\overline{C} + \overline{D}) = A\overline{B}(\overline{C} + \overline{D})$

(c) $\overline{(A + B)\overline{C}\,\overline{D} + E + \overline{F}} = \overline{[(A + B)\overline{C}\,\overline{D}]}\;\overline{(E + \overline{F})} = (\overline{A}\,\overline{B} + C + D)\overline{E}F$

Problema relacionado. Aplicar los teoremas de DeMorgan a la expresión $\overline{\overline{A}B(C + \overline{D}) + E}$

EJEMPLO 4.7 La expresión booleana de una puerta OR-exclusiva es $A\overline{B} + \overline{A}B$. Tomando esto como punto de partida, desarrollar una expresión para una puerta NOR-exclusiva, utilizando los teoremas de DeMorgan y aquellas leyes o reglas que se puedan aplicar.

Solución

En primer lugar, se complementa la expresión OR-exclusiva, y luego se aplican los teoremas de DeMorgan del siguiente modo:

$$\overline{A\overline{B} + \overline{A}B} = (\overline{A\overline{B}})(\overline{\overline{A}B}) = (\overline{A} + \overline{\overline{B}})(\overline{\overline{A}} + \overline{B}) = (\overline{A} + B)(A + \overline{B})$$

A continuación, se aplica la ley distributiva y la regla 8 ($A \cdot \overline{A} = 0$):

$$(\overline{A} + B)(A + \overline{B}) = \overline{A}A + \overline{A}\,\overline{B} + AB + B\overline{B} = \overline{A}\,\overline{B} + AB$$

La expresión resultante para una puerta XNOR es $\overline{A}\,\overline{B} + AB$. Esta expresión es igual a 1 siempre que ambas variables son 0 o 1.

Problema relacionado. Utilizando la expresión para una puerta NAND de 4 entradas, utilizar los teoremas de DeMorgan para desarrollar una expresión para una puerta negativa-OR de 4 entradas.

REVISIÓN DE LA SECCIÓN 4.3

1. Aplicar los teoremas de DeMorgan a las siguientes expresiones:
   (a) $\overline{ABC} + (\overline{D} + \overline{E})$ (b) $\overline{(A+B)C}$ (c) $\overline{A+B+C} + \overline{DE}$

4.4 ■ ANÁLISIS BOOLEANO DE LOS CIRCUITOS LÓGICOS

El álgebra de Boole proporciona una manera concisa de expresar el funcionamiento de un circuito lógico formado por una combinación de puertas lógicas, de tal forma que la salida puede determinarse por la combinación de los valores de entrada.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Determinar las expresiones booleanas de una combinación de puertas. ■ Determinar el funcionamiento lógico de un circuito a partir de su expresión booleana. ■ Elaborar una tabla de verdad.

Expresión booleana de un circuito lógico

Para obtener la expresión booleana de un determinado circuito lógico, la manera de proceder consiste en comenzar con las entradas situadas más a la izquierda e ir avanzando hasta las líneas de salida, escribiendo la expresión para cada puerta. Para el circuito ejemplo de la Figura 4.16, su expresión booleana se determina de la siguiente manera:

1. La expresión de la puerta AND situada más a la izquierda cuyas entradas son C y D es CD.
2. La salida de la puerta AND situada más a la izquierda es una de las entradas de la puerta OR y B es su otra entrada. Por tanto, la expresión para la puerta OR es $B + CD$.
3. La salida de la puerta OR es una de las entradas de la puerta AND situada más a la derecha, siendo A su otra entrada. Por tanto, la expresión de esta puerta AND será $A(B + CD)$, que es la expresión final de salida del circuito completo.

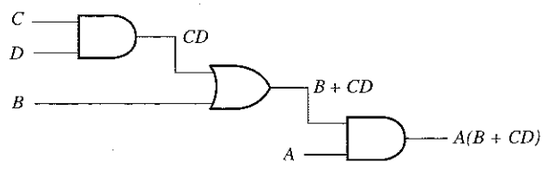


FIGURA 4.16
Circuito lógico que muestra el desarrollo de la expresión booleana para la salida.

Elaboración de la tabla de verdad de un circuito lógico

Una vez que se ha determinado la expresión booleana de un circuito dado, puede desarrollarse una tabla de verdad que represente la salida del circuito lógico para todos los valores posibles de las variables de entrada. Esto requiere que se evalúe la expresión booleana para todas las posibles combinaciones de valores de las variables de entrada. En el caso del circuito de la Figura 4.16, existen cuatro variables de entrada (A, B, C y D) y, por tanto, hay $2^4 = 16$ posibles combinaciones de valores.

Evaluación de la expresión. Para evaluar la expresión $A(B + CD)$, en primer lugar, utilizando las reglas de la adición y multiplicación booleanas, se localizan los valores de las variables que hacen que la expresión sea igual a 1. En este caso, la expresión es igual a 1 sólo si $A = 1$ y $(B + CD) = 1$, ya que:

$$A(B + CD) = 1 \cdot 1 = 1$$

Ahora, se determina cuándo el término $B + CD$ es igual a 1. La expresión $B + CD$ es 1 si $B = 1$ o $CD = 1$, o ambos son igual a 1, ya que:

$$B + CD = 1 + 0 = 1$$
$$B + CD = 0 + 1 = 1$$
$$B + CD = 1 + 1 = 1$$

El término CD es 1 sólo si C y D son 1.

Para resumir, la expresión $A(B + CD) = 1$ cuando $A = 1$ y $B = 1$, independientemente del valor de C y D, o cuando $A = 1$ y $C = 1$ y $D = 1$, independientemente del valor de B. La expresión $A(B + CD)$ vale 0 para el resto de combinaciones posibles de las variables.

Representación de los resultados en una tabla de verdad. El primer paso consiste en enumerar las dieciséis combinaciones de unos y ceros de las variables de entrada en una secuencia binaria, como muestra la Tabla 4.5. A continuación, se pone un 1 en la columna de salida para las combinaciones de variables de entrada que se han determinado en la evaluación de la expresión. Finalmente, se escribe un 0 en la columna de salida para el resto de las combinaciones de las variables de entrada. Estos resultados se muestran en la Tabla 4.5.

TABLA 4.5
Tabla de verdad del circuito lógico de la Figura 4.16.

Entradas				Salida	Entradas				Salida
A	*B*	*C*	*D*	*A(B + CD)*	*A*	*B*	*C*	*D*	*A(B + CD)*
0	0	0	0	0	1	0	0	0	0
0	0	0	1	0	1	0	0	1	0
0	0	1	0	0	1	0	1	0	0
0	0	1	1	0	1	0	1	1	1
0	1	0	0	0	1	1	0	0	1
0	1	0	1	0	1	1	0	1	1
0	1	1	0	0	1	1	1	0	1
0	1	1	1	0	1	1	1	1	1

REVISIÓN DE LA SECCIÓN 4.4

1. Reemplazar las puertas AND por puertas OR y viceversa en la Figura 4.16, y determinar la expresión booleana de salida.
2. Elaborar la tabla de verdad del circuito de la cuestión 1.

4.5 ■ SIMPLIFICACIÓN MEDIANTE EL ÁLGEBRA DE BOOLE

Muchas veces, a la hora de aplicar el álgebra booleana, hay que reducir una expresión a su forma más simple o cambiarla a una forma más conveniente para conseguir una implementación más eficiente. El método que se va a tratar en esta sección utiliza las reglas, leyes y teoremas del álgebra de Boole para manipular y simplificar una expresión. Este método requiere un profundo conocimiento del álgebra booleana y una considerable experiencia en su aplicación, por no mencionar también un poquito de ingenio y destreza.

Al finalizar esta sección, el lector deberá ser capaz de:

- Aplicar las leyes, reglas y teoremas del álgebra de Boole para simplificar cualquier expresión.

Una expresión booleana simplificada emplea el menor número posible de puertas en la implementación de una determinada expresión. A continuación, se presentan cuatro ejemplos de simplificación descritos paso a paso.

EJEMPLO 4.8 Mediante las técnicas del álgebra de Boole, simplificar la siguiente expresión:

$$AB + A(B + C) + B(B + C)$$

Solución

El método que se aplica no es necesariamente el único posible.

Paso 1. Aplicar la ley distributiva al segundo y tercer término de la expresión del siguiente modo:

$$AB + AB + AC + BB + BC$$

Paso 2. Aplicar la regla 7 ($BB = B$) al cuarto término:

$$AB + AB + AC + B + BC$$

Paso 3. Aplicar la regla 5 ($AB + AB = AB$) a los dos primeros términos:

$$AB + AC + B + BC$$

Paso 4. Aplicar la regla 10 ($B + BC = B$) a los dos últimos términos:

$$AB + AC + B$$

Paso 5. Aplicar la regla 10 ($AB + B = B$) a los términos primero y tercero:

$$B + AC$$

En este instante, la expresión ya no puede seguir simplificándose. Según se vaya adquiriendo experiencia en la aplicación del álgebra de Boole, se podrán combinar muchos pasos individuales.

Problema relacionado. Simplificar la expresión $A\overline{B} + A(\overline{B+C}) + B(\overline{B+C})$.

La Figura 4.17 muestra cómo el proceso de simplificación del Ejemplo 4.8 ha reducido significativamente el número de puertas lógicas necesarias para implementar la expresión. En la parte (a) se puede ver que son necesarias cinco puertas para implementar dicha expresión en su forma original, mientras que sólo se requieren dos para hacerlo una vez simplificada, como se muestra en la parte (b). Es importante resaltar que estas dos redes de puertas son equivalentes, es decir, para cualquier combinación de valores en las entradas A, B y C, obtenemos siempre la misma salida en ambos circuitos.

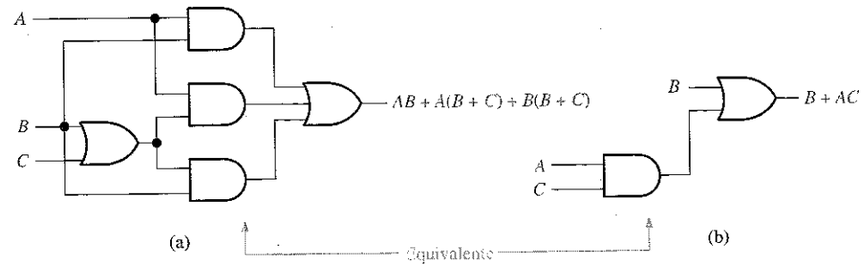


FIGURA 4.17
Redes de puertas para el Ejemplo 4.8.

EJEMPLO 4.9 Simplificar la expresión booleana:

$$[A\overline{B}(C + BD) + \overline{A}\,\overline{B}]C$$

Los corchetes y paréntesis significan lo mismo: el término de su interior se multiplica (AND) por el término exterior.

Solución

Paso 1. Aplicar la ley distributiva a los términos entre corchetes:

$$(A\overline{B}C + A\overline{B}BD + \overline{A}\,\overline{B})C$$

Paso 2. Aplicar la regla 8 ($\overline{B}B = 0$) al segundo término entre paréntesis:

$$(A\overline{B}C + A \cdot 0 \cdot D + \overline{A}\,\overline{B})C$$

Paso 3. Aplicar la regla 3 ($A \cdot 0 \cdot D = 0$) al segundo término:

$$(A\overline{B}C + 0 + \overline{A}\,\overline{B})C$$

Paso 4. Aplicar la regla 1 (quitar el 0) dentro del paréntesis:

$$(A\overline{B}C + \overline{A}\,\overline{B})C$$

Paso 5. Aplicar la ley distributiva:

$$A\overline{B}CC + \overline{A}\,\overline{B}C$$

Paso 6. Aplicar la regla 7 ($CC = C$) al primer término:

$$A\overline{B}C + \overline{A}\,\overline{B}C$$

Paso 7. Sacar factor común $\overline{B}C$:

$$\overline{B}C(A + \overline{A})$$

Paso 8. Aplicar la regla 6 ($A + \overline{A} = 1$):

$$\overline{B}C \cdot 1$$

Paso 9. Aplicar la regla 4 (quitar el 1):

$$\overline{B}C$$

Problema relacionado. Simplificar la expresión $[AB(C + \overline{BD}) + \overline{AB}]CD$.

EJEMPLO 4.10 Simplificar la expresión booleana:

$$\overline{A}BC + A\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,\overline{C} + A\overline{B}C + ABC$$

Solución

Paso 1. Sacar factor común BC del primer y último término:

$$BC(\overline{A} + A) + A\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,\overline{C} + A\overline{B}C$$

Paso 2. Aplicar la regla 6 ($\overline{A} + A = 1$) al término entre paréntesis y sacar el factor común $A\overline{B}$ del segundo y último término:

$$BC \cdot 1 + A\overline{B}(\overline{C} + C) + \overline{A}\,\overline{B}\,\overline{C}$$

Paso 3. Aplicar la regla 4 (quitar el 1) al primer término y la regla 6 ($\overline{C} + C = 1$) al término entre paréntesis:

$$BC + A\overline{B} \cdot 1 + \overline{A}\,\overline{B}\,\overline{C}$$

Paso 4. Aplicar la regla 4 (quitar el 1) al segundo término:

$$BC + A\overline{B} + \overline{A}\,\overline{B}\,\overline{C}$$

Paso 5. Sacar factor común $\overline{B}$ del segundo y tercer término:

$$BC + \overline{B}(A + \overline{A}\,\overline{C})$$

Paso 6. Aplicar la regla 11 ($A + \overline{A}\,\overline{C} = A + \overline{C}$) al término entre paréntesis:

$$BC + \overline{B}(A + \overline{C})$$

Paso 7. Usar las leyes distributiva y conmutativa para obtener la siguiente expresión:

$$BC + A\overline{B} + \overline{B}\,\overline{C}$$

Problema relacionado. Simplificar la expresión $AB\overline{C} + \overline{A}\,\overline{B}C + \overline{A}BC + \overline{A}\,\overline{B}\,\overline{C}$.

EJEMPLO 4.11 Simplificar la siguiente expresión:

$$\overline{AB + AC} + \overline{A}\,\overline{B}C$$

Solución

Paso 1. Aplicar el teorema de DeMorgan al primer término:

$$(\overline{AB})(\overline{AC}) + \overline{A}\,\overline{B}C$$

Paso 2. Aplicar el teorema de DeMorgan a cada término entre paréntesis:

$$(\overline{A} + \overline{B})(\overline{A} + \overline{C}) + \overline{A}\,\overline{B}C$$

Paso 3. Aplicar la ley distributiva a los dos términos entre paréntesis:

$$\overline{A}\,\overline{A} + \overline{A}\,\overline{C} + \overline{A}\,\overline{B} + \overline{B}\,\overline{C} + \overline{A}\,\overline{B}C$$

Paso 4. Aplicar la regla 7 ($\overline{A}\,\overline{A} = \overline{A}$) al primer término y la regla 10 [$\overline{A}\,\overline{B} + \overline{A}\,\overline{B}C = \overline{A}\,\overline{B}(1 + C) = \overline{A}\,\overline{B}$] al tercer y al último término:

$$\overline{A} + \overline{A}\,\overline{C} + \overline{A}\,\overline{B} + \overline{B}\,\overline{C}$$

Paso 5. Aplicar la regla 10 [$\overline{A} + \overline{A}\,\overline{C} = \overline{A}(1 + \overline{C}) = \overline{A}$] a los términos primero y segundo:

$$\overline{A} + \overline{A}\,\overline{B} + \overline{B}\,\overline{C}$$

Paso 6. Aplicar la regla 10 [$\overline{A} + \overline{A}\,\overline{B} = \overline{A}(1 + \overline{B}) = \overline{A}$] a los términos primero y segundo:

$$\overline{A} + \overline{B}\,\overline{C}$$

Problema relacionado. Simplificar la expresión booleana $\overline{AB} + \overline{AC} + \overline{A}\,\overline{B}\,\overline{C}$.

REVISIÓN DE LA SECCIÓN 4.5

1. Simplificar, si es posible, las siguientes expresiones booleanas:
   (a) $A + AB + ABC$ **(b)** $(\overline{A} + B)C + ABC$ **(c)** $A\overline{B}C(BD + CDE) + A\overline{C}$
2. Implementar con las puertas lógicas apropiadas cada expresión de la cuestión anterior. Después, implementar la expresión simplificada y comparar el número de puertas empleado en cada caso.

4.6 ■ FORMAS ESTÁNDAR DE LAS EXPRESIONES BOOLEANAS

Todas las expresiones booleanas, independientemente de su forma, pueden convertirse en cualquiera de las dos formas estándar: suma de productos o producto de sumas. Esto posibilita que la evaluación, simplificación e implementación de las expresiones booleanas sea mucho más sistemática y sencilla.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Identificar una expresión en forma de suma de productos. ■ Determinar el dominio de una expresión booleana. ■ Convertir cualquier suma de productos a su forma

estándar. ■ Evaluar una expresión en forma de suma de productos según los valores binarios. ■ Identificar una expresión en forma de producto de sumas. ■ Convertir cualquier producto de sumas a su forma estándar. ■ Evaluar una expresión en forma de producto de sumas según los valores binarios. ■ Convertir expresiones de una a otra forma estándar.

Suma de productos

En la Sección 4.1, se ha definido el término producto como un término que es el producto (multiplicación booleana) de literales (variables o sus complementos). Cuando dos o más productos se suman mediante la adición booleana, la expresión resultante se denomina **suma de productos (SOP, Sum Of Products)**. Algunos ejemplos son:

$$AB + ABC$$
$$ABC + CDE + \overline{B}C\overline{D}$$
$$\overline{A}B + \overline{A}B\overline{C} + AC$$

Una suma de productos puede también contener términos de una única variable, como $A + \overline{A}\,\overline{B}C + BC\overline{D}$. Si volvemos a los ejemplos de simplificación de la última sección, puede observarse que cada término de la expresión resultante era o un producto aislado o una suma de productos. En una expresión con formato suma de productos, una barra no puede extenderse sobre más de una variable, aunque más de una variable pueda tener una barra encima. Por ejemplo, una suma de productos puede contener el término $\overline{A}\,\overline{B}\,\overline{C}$, pero no el $\overline{ABC}$.

Dominio de una expresión booleana. El **dominio** de una expresión booleana es el conjunto de variables (o sus complementos) contenido en una expresión. Por ejemplo, el dominio de la expresión $\overline{A}B + A\overline{B}C$ es el conjunto de variables *A, B, C;* y el dominio de la expresión $AB\overline{C} + C\overline{D}E + \overline{B}C\overline{D}$ es el conjunto *A, B, C, D, E.*

Implementación de una suma de productos. La implementación de una suma de productos simplemente requiere aplicar la operación OR a las salidas de dos o más puertas AND. Una operación AND da lugar a un producto, y la adición de dos o más productos se realiza mediante puertas OR. Por tanto, una expresión suma de productos puede implementarse mediante un circuito lógico AND-OR en el que las salidas de las puertas AND, cuyo número es igual al de productos que contenga la expresión, son las entradas de una puerta OR, como se muestra en la Figura 4.18 para la expresión *AB+BCD+AC*. La salida *X* de la puerta OR es igual a la suma de productos.

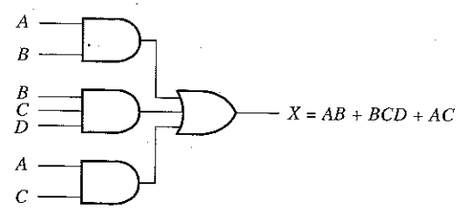


FIGURA 4.18
Implementación de la suma de productos AB + BCD + AC.

Conversión de una expresión general a formato suma de productos

Cualquier expresión lógica puede ser transformada a una expresión suma de productos, aplicando el álgebra de Boole. Por ejemplo, la expresión $A(B + CD)$ puede convertirse en una suma de productos aplicando la ley distributiva:

$$A\,(B + CD) = AB + ACD$$

EJEMPLO 4.12 Convertir cada una de las siguientes expresiones booleanas a su forma suma de productos:

(a) $AB + B(CD + EF)$ **(b)** $(A + B)(B + C + D)$ **(c)** $\overline{(A+B) + C}$

Solución

(a) $AB + B(CD + EF) = AB + BCD + BEF$

(b) $(A + B)(B + C + D) = AB + AC + AD + BB + BC + BD$

(c) $\overline{(A+B) + C} = \overline{(A+B)}\overline{C} = (A + B)\overline{C} = A\overline{C} + B\overline{C}$

Problema relacionado. Convertir $\overline{A}B\overline{C} + (A + \overline{B})(B + \overline{C} + A\overline{B})$ a la forma suma de productos.

Forma estándar de la suma de productos

Hasta ahora, hemos estado viendo sumas de productos en las que algunos de los términos no contenían todas las variables del dominio de la expresión. Por ejemplo, la expresión $\overline{A}B\overline{C} + A\overline{B}D + \overline{A}B\overline{C}D$ tiene un dominio formado por las variables *A, B, C* y *D*. Podemos ver que el conjunto completo de variables del dominio no está incluido en los dos primeros términos de la expresión, es decir, faltan *D* o $\overline{D}$ en el primer término y *C* o $\overline{C}$ en el segundo.

Una suma de productos estándar es aquélla en la que *todas* las variables del dominio aparecen en cada uno de los términos de la expresión. Por ejemplo, $A\overline{B}CD + \overline{A}\,\overline{B}C\overline{D} + AB\overline{C}\,\overline{D}$ es una suma de productos estándar. La suma de productos estándar tienen mucha importancia a la hora de construir las tablas de verdad, que se tratarán en la Sección 4.7, junto con el método de simplificación de los mapas de Karnaugh, que se verá en la Sección 4.8. Cualquier suma de productos en forma no estándar (que denominaremos simplemente suma de productos) puede convertirse al formato estándar utilizando el álgebra de Boole.

Conversión de suma de productos a su forma estándar. Cada término producto de una suma de productos que no contenga todas las variables del dominio, puede ser transformada a su forma estándar de manera que incluya todas las variables del dominio o sus complementos. Como se muestra a continuación, una suma de productos puede convertirse en forma estándar mediante la regla 6 del álgebra booleana ($A + \overline{A} = 1$) de la Tabla 4.1, que establece que una variable sumada con su complemento es siempre igual a 1.

Paso 1. Multiplicar cada término producto no estándar por un término formado por la suma de la variable que falta y su complemento. Con esto se obtienen dos términos producto. Como se sabe, se puede multiplicar por 1 cualquier expresión sin que se altere su valor.

Paso 2. Repetir el paso 1 hasta que todos los términos de la expresión contengan todas las variables (o sus complementos) del dominio. Al convertir cada producto a su forma estándar, el número de términos producto se duplica por cada variable que falta, como muestra el Ejemplo 4.13.

EJEMPLO 4.13 Convertir la siguiente expresión booleana al formato suma de productos estándar:

$$A\overline{B}C + \overline{A}\,\overline{B} + AB\overline{C}D$$

Solución

El dominio de esta suma de productos es *A, B, C, D.* Considerando cada término por separado, se comprueba que al primer término, $A\overline{B}C$, le falta la variable D o $\overline{D}$, por lo que lo multiplicamos por D o $\overline{D}$, obteniendo:

$$A\overline{B}C = A\overline{B}C(D + \overline{D}) = A\overline{B}CD + A\overline{B}C\overline{D}$$

En este caso, se obtienen dos productos estándar. En el segundo término, $\overline{A}\,\overline{B}$, faltan las variables C o $\overline{C}$ y D o $\overline{D}$, de manera que multiplicamos primero por $C + \overline{C}$:

$$\overline{A}\,\overline{B} = \overline{A}\,\overline{B}(C + \overline{C}) = \overline{A}\,\overline{B}C + \overline{A}\,\overline{B}\,\overline{C}$$

Los dos términos que obtenemos carecen de la variable D o $\overline{D}$, por lo que multiplicamos por $D + \overline{D}$:

$$\begin{aligned}\overline{A}\,\overline{B} &= \overline{A}\,\overline{B}C + \overline{A}\,\overline{B}\,\overline{C} \\ &= \overline{A}\,\overline{B}C(D + \overline{D}) + \overline{A}\,\overline{B}\,\overline{C}(D + \overline{D}) \\ &= \overline{A}\,\overline{B}CD + \overline{A}\,\overline{B}C\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}\end{aligned}$$

En este caso, el resultado son cuatro productos estándar. El tercer término $AB\overline{C}D$, ya está en formato estándar. La suma de productos estándar que obtenemos es finalmente:

$$A\overline{B}C + \overline{A}\,\overline{B} + AB\overline{C}D = A\overline{B}CD + A\overline{B}C\overline{D} + \overline{A}\,\overline{B}CD + \overline{A}\,\overline{B}C\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + AB\overline{C}D$$

Problema relacionado. Convertir la expresión $W\overline{X}Y + \overline{X}Y\overline{Z} + WX\overline{Y}$ en una suma de productos estándar.

Representación binaria de un término producto estándar. Un término producto estándar es igual a 1 sólo para una combinación de los valores de las variables. Por ejemplo, el término $A\overline{B}C\overline{D}$ es igual a 1 cuando $A = 1$, $B = 0$, $C = 1$ y $D = 0$, como se muestra más abajo, y es cero para el resto de combinaciones de valores de las variables.

$$A\overline{B}C\overline{D} = 1 \cdot \overline{0} \cdot 1 \cdot \overline{0} = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

En este caso, el término producto tiene un valor binario de 1010 (diez en decimal).

Recordemos que un producto se implementa mediante una puerta AND, cuya salida es 1 si y sólo si todas sus entradas son 1. Para generar el complemento de una variable se utilizan inversores.

Una suma de productos (SOP) es igual a 1 si y sólo si uno o más de los términos producto que forman la expresión es igual a 1.

EJEMPLO 4.14 Determinar los valores binarios para los que la siguiente suma de productos estándar sea igual a 1:

$$ABCD + A\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$$

Solución

El término $ABCD$ es igual a 1 si $A = 1$, $B = 1$, $C = 1$ y $D = 1$.

$$ABCD = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

El término $A\overline{B}\,\overline{C}D$ es igual a 1 si $A = 1$, $B = 0$, $C = 0$ y $D = 1$.

$$A\overline{B}\,\overline{C}D = 1 \cdot \overline{0} \cdot \overline{0} \cdot 1 = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

El término $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$ es igual a 1 cuando $A = 0$, $B = 0$, $C = 0$ y $D = 0$.

$$\overline{A}\,\overline{B}\,\overline{C}\,\overline{D} = \overline{0} \cdot \overline{0} \cdot \overline{0} \cdot \overline{0} = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

La suma de productos es igual a 1 sólo cuando cualquiera de los tres términos, o todos, son igual a 1.

Problema relacionado. Determinar los valores binarios para los que la siguiente suma de productos es 1:

$$\overline{X}YZ + X\overline{Y}Z + XY\overline{Z} + \overline{X}\overline{Y}\overline{Z} + XYZ$$

¿Es ésta una suma de productos estándar?

Producto de sumas

En la Sección 4.1 se ha definido el término suma como un término formado por la suma (adición booleana) de literales (variables o sus complementos). Cuando dos o más términos suma se multiplican, la expresión resultante es un **producto de sumas (POS, Product Of Sums)**. Algunos ejemplos son:

$$(\overline{A} + B)(A + \overline{B} + C)$$
$$(\overline{A} + \overline{B} + \overline{C})(C + \overline{D} + E)(\overline{B} + C + D)$$
$$(A + B)(A + \overline{B} + C)(\overline{A} + C)$$

Un producto de sumas puede contener términos con una única variable, como en $\overline{A}$ $(A + \overline{B} + C)$ $(\overline{B} + \overline{C} + D)$. En un producto de sumas, una barra no puede extenderse nunca sobre más de una variable, aunque más de una variable puede tener una barra encima. Por ejemplo, un producto de sumas puede contener el término $\overline{A} + \overline{B} + \overline{C}$ pero no el $\overline{A + B + C}$.

Implementación de un producto de sumas. La implementación de un producto de sumas requiere simplemente la aplicación de la operación AND a las salidas de dos o más puertas OR. Un sumando se origina mediante la operación OR, y el producto de varios términos suma se realiza por medio de la operación AND. Por consiguiente, un producto de sumas puede implementarse a partir de puertas lógicas OR (cuyo número será igual al de sumandos de la expresión) cuyas salidas se conectan a las entradas de una puerta AND, como muestra la Figura 4.19 para la expresión $(A+B)(B+C+D)(A+C)$. La salida X de la puerta AND es igual al producto de sumas.

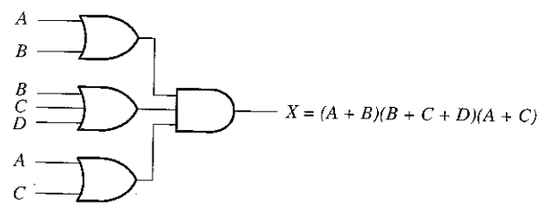


FIGURA 4.19
Implementación del producto de sumas $(A + B)(B + C + D)(A + C)$.

Forma estándar del producto de sumas

Hasta ahora, se han tratado expresiones producto de sumas en las que algunos de los términos no contenían todas las variables del dominio de la expresión. Por ejemplo, la expresión:

$$(A + \overline{B} + C)(A + B + \overline{D})(A + \overline{B} + \overline{C} + D)$$

tiene un dominio formado por las variables A, B, C y D. Obsérvese que el conjunto completo de variables del dominio no está representado en los dos primeros términos de la expresión; es decir, faltan D o $\overline{D}$ en el primer término, y C o $\overline{C}$ en el segundo término.

Un producto de sumas estándar es aquel en el que *todas* las variables del dominio o sus complementos aparecen en cada uno de los términos de la expresión. Por ejemplo;

$$(\overline{A} + \overline{B} + \overline{C} + \overline{D})(A + \overline{B} + C + D)(A + B + \overline{C} + D)$$

es un producto de sumas estándar. Cualquier producto de sumas no estándar (que denominaremos simplemente producto de sumas) puede convertirse a su forma estándar mediante el álgebra de Boole.

Conversión de un producto de sumas a su forma estándar. Cada operación suma de un producto de sumas que no contenga todas las variables del dominio puede pasarse a formato estándar, de manera que incluya todas las variables del dominio o sus complementos. Como se muestra a continuación, un producto de sumas no estándar se pasa a formato estándar utilizando la regla booleana número 8 ($A \cdot \overline{A} = 0$) de la Tabla 4.1, la cual establece que una variable multiplicada por su complemento es igual a 0.

Paso 1. Añadir a cada "término suma" no estándar un término consistente en el producto de la variable que falta y su complemento; esto da lugar a la aparición de dos sumandos en la expresión. Como ya se sabe, siempre se puede sumar 0 sin que se altere el valor de una expresión.

Paso 2. Aplicar la regla 12 de la Tabla 4.1: $A + BC = (A + B)(A + C)$.

Paso 3. Repetir el paso 1 hasta que todos los sumandos resultantes contengan todas las variables del dominio o sus complementos.

EJEMPLO 4.15 Convertir la siguiente expresión booleana a formato producto de sumas estándar:

$$(A + \overline{B} + C)(\overline{B} + C + \overline{D})(A + \overline{B} + \overline{C} + D)$$

Solución

El dominio de esta expresión es A, B, C, y D. Vamos a considerar término por término. El primero, $A + \overline{B} + C$, no contiene la variable D o $\overline{D}$, por lo que añadimos $D\overline{D}$ y aplicamos la regla 12 como sigue:

$$A + \overline{B} + C = A + \overline{B} + C + D\overline{D} = (A + \overline{B} + C + D)(A + \overline{B} + C + \overline{D})$$

El segundo término, $\overline{B} + C + \overline{D}$, carece de la variable A o $\overline{A}$, por lo que añadimos $A\overline{A}$ y aplicamos la regla 12 del siguiente modo:

$$\overline{B} + C + \overline{D} = \overline{B} + C + \overline{D} + A\overline{A} = (A + \overline{B} + C + \overline{D})(\overline{A} + \overline{B} + C + \overline{D})$$

El tercer término, $A + \overline{B} + \overline{C} + D$, ya está escrito en su forma estándar. El producto de sumas estándar de la expresión original es:

$$(A + \overline{B} + C)(\overline{B} + C + \overline{D})(A + \overline{B} + \overline{C} + D) =$$
$$(A + \overline{B} + C + D)(A + \overline{B} + C + \overline{D})(A + \overline{B} + C + \overline{D})(\overline{A} + \overline{B} + C + \overline{D})(A + \overline{B} + \overline{C} + D)$$

Problema relacionado. Convertir la expresión $(A + \overline{B})(B + C)$ a su forma producto de sumas estándar.

Representación binaria de un término suma estándar. Un término suma estándar es igual a 0 sólo para una única combinación de los valores de las variables. Por ejemplo, la suma $A + \overline{B} + C + \overline{D}$ es 0 cuando $A = 0$, $B = 1$, $C = 0$ y $D = 1$, como se muestra más abajo, y es 1 para el resto de combinaciones de los valores de las variables.

$$A + \overline{B} + C + \overline{D} = 0 + \overline{1} + 0 + \overline{1} = 0 + 0 + 0 + 0 = 0$$

En este caso, el valor binario de la suma es 0101 (5 en decimal). Recuerde que una suma se implementa mediante una puerta OR cuya salida es 0 si y sólo si cada una de sus entradas es 0. Para complementar las variables se utilizan inversores.

Un producto de sumas (POS) es igual a 0 si y sólo si uno o más términos suma de la expresión son igual a 0.

EJEMPLO 4.16 Determinar los valores binarios de las variables para los que la siguiente expresión producto de sumas estándar es igual a 0:

$$(A + B + C + D)(A + \overline{B} + \overline{C} + D)(\overline{A} + \overline{B} + \overline{C} + \overline{D})$$

Solución

El término $A + B + C + D$ es igual a 0 cuando $A = 0$, $B = 0$, $C = 0$ y $D = 0$.

$$A + B + C + D = 0 + 0 + 0 + 0 = 0$$

El término $A + \overline{B} + \overline{C} + D$ es igual a 0 si $A = 0$, $B = 1$, $C = 1$ y $D = 0$.

$$A + \overline{B} + \overline{C} + D = 0 + \overline{1} + \overline{1} + 0 = 0 + 0 + 0 + 0 = 0$$

El término $\overline{A} + \overline{B} + \overline{C} + \overline{D}$ es igual a 0 si $A = 1$, $B = 1$, $C = 1$ y $D = 1$.

$$\overline{A} + \overline{B} + \overline{C} + \overline{D} = \overline{1} + \overline{1} + \overline{1} + \overline{1} = 0 + 0 + 0 + 0 = 0$$

El producto de sumas es igual a 0 cuando cualquiera de los tres términos suma es igual a 0.

Problema relacionado. Determinar los valores binarios para los que el siguiente producto de sumas es igual a 0:

$$(X + \overline{Y} + Z)(\overline{X} + Y + Z)(X + Y + \overline{Z})(\overline{X} + \overline{Y} + \overline{Z})(X + \overline{Y} + \overline{Z})$$

¿Es esta expresión un producto de sumas estándar?

Conversión de una suma de productos estándar a producto de sumas estándar

Los valores binarios de los términos producto en una suma de productos estándar dada no aparecen en su producto de sumas estándar equivalente. Asimismo, los valores binarios que no están representados en una suma de productos sí aparecen en el producto de sumas equivalente. Por lo tanto, para pasar de la suma de productos estándar al producto de sumas estándar hay que realizar los siguientes pasos:

Paso 1. Evaluar cada término de la expresión suma de productos; es decir, determinar los valores binarios que representan estos términos.

Paso 2. Determinar todos los números binarios no incluidos al realizar el cálculo del paso 1.

Paso 3. Escribir los términos suma equivalentes para cada valor binario del paso 2 y expresarlos en forma de producto de sumas.

Mediante un procedimiento similar se puede pasar de un producto de sumas a una suma de productos.

EJEMPLO 4.17 Convertir la siguiente expresión suma de productos en su expresión equivalente como producto de sumas:

$$\overline{A}\,\overline{B}\,\overline{C} + \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}C + ABC$$

Solución

El resultado de la evaluación es el siguiente:

$$000 + 010 + 011 + 101 + 111$$

Puesto que son tres las variables que conforman el dominio de esta expresión, existe un total de ocho (2^3) posibles combinaciones. La expresión suma de productos contiene cinco de estas combinaciones, luego la expresión producto de sumas debe contener las otras tres, que son 001, 100 y 110. Recuerde que estos son los valores binarios que hacen que cada operación suma sea igual a cero. El producto de sumas equivalente es:

$$(A + B + \overline{C})(\overline{A} + B + C)(\overline{A} + \overline{B} + C)$$

Problema relacionado. Sustituyendo los valores binarios en cada término, verificar que las expresiones suma de productos y producto de sumas de este ejemplo son equivalentes.

REVISIÓN DE LA SECCIÓN 4.6

1. Determinar si cada una de las siguientes expresiones es una suma de productos o un producto de sumas. Indicar si es o no una forma estándar.
   (a) $AB + \overline{A}BD + \overline{A}C\overline{D}$ **(b)** $(A + \overline{B} + C)(A + B + \overline{C})$
   (c) $\overline{A}BC + ABC$ **(d)** $A(A + \overline{C})(A + B)$
2. Convertir las sumas de productos de la cuestión 1 a su forma estándar.
3. Convertir los productos de sumas de la cuestión 1 a su forma estándar.

4.7 ■ EXPRESIONES BOOLEANAS Y TABLAS DE VERDAD

Todas las expresiones booleanas pueden ser fácilmente convertidas en tablas de verdad utilizando los valores binarios de cada término de la expresión. La tabla de verdad es una forma muy común, en un formato muy conciso, de expresar el funcionamiento lógico de un circuito. Además, las expresiones suma de productos y producto de sumas pueden calcularse mediante las tablas de verdad. Las tablas de verdad se pueden encontrar en las hojas de especificaciones y en otras documentaciones relativas al funcionamiento de los circuitos y sistemas digitales.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Pasar una suma de productos a su tabla de verdad. ■ Pasar un producto de sumas a su tabla de verdad. ■ Obtener una expresión estándar a partir de una tabla de verdad. ■ Interpretar correctamente los datos contenidos en una tabla de verdad.

Conversión de una suma de productos a tabla de verdad

Como se ha establecido en la Sección 4.6, una suma de productos es igual a 1 si y sólo si al menos uno de los productos es igual a 1. Una tabla de verdad es sencillamente una lista de las posibles combinaciones de valores de las variables de entrada y sus correspondientes valores de salida (1 o 0). Para una expresión cuyo dominio es de dos variables, existen cuatro combinaciones distintas de estas variables ($2^2 = 4$). Para una expresión cuyo dominio consiste en tres variables, existen ocho ($2^3 = 8$) combinaciones diferentes de estas variables. Para una expresión cuyo

dominio contiene cuatro variables, el número de combinaciones es ($2^4 = 16$), y así sucesivamente.

El primer paso a seguir para elaborar una tabla de verdad es enumerar todas las posibles combinaciones de los valores de las variables de la expresión. A continuación, hay que pasar la suma de productos a su formato estándar, si no lo está ya. Por último, se escribe un 1 en la columna de salida (X) para cada valor binario que hace que la suma de productos estándar sea 1, y se escribe un 0 para los restantes valores. Este procedimiento se muestra en el Ejemplo 4.18.

EJEMPLO 4.18 Desarrollar una tabla de verdad para la expresión suma de productos estándar $\overline{A}\,\overline{B}C + A\overline{B}\,\overline{C} + ABC$.

Solución

Existen tres variables en el dominio, de forma que hay ocho posibles combinaciones de valores binarios de las variables, como se muestra en las tres primeras columnas de la Tabla 4.6. Los valores binarios que hacen que los productos de la expresión sean 1 son $\overline{A}\,\overline{B}C$: 001; $A\overline{B}\,\overline{C}$: 100 y ABC: 111; Para cada uno de estos valores binarios, se escribe un 1 en la columna de salida como se muestra en la tabla. Para cada una de las restantes combinaciones, se coloca un 0 en la columna de salida.

TABLA 4.6

Entradas			Salida
A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Problema relacionado. Crear una tabla de verdad para la expresión $\overline{A}B\overline{C} + A\overline{B}C$.

Conversión de un producto de sumas a tabla de verdad

Recordemos que un producto de sumas es igual a 0 solamente si al menos uno de los términos suma es igual a 0. Para elaborar la tabla de verdad de un producto de sumas, se enumeran todas las posibles combinaciones de valores binarios de las variables, del mismo modo que se ha hecho anteriormente para la suma de productos. A continuación, se pasa la expresión producto de sumas a su formato estándar, si todavía no lo está. Por último, se escribe un 0 en la columna de salida (X) para cada valor binario que hace que la expresión sea cero, y se escribe un 1 en el resto de las posibles combinaciones. Este procedimiento se ilustra en el Ejemplo 4.19.

EJEMPLO 4.19 Determinar la Tabla de verdad de la siguiente expresión producto de sumas estándar:

$$(A + B + C)(A + \overline{B} + C)(A + \overline{B} + \overline{C})(\overline{A} + B + \overline{C})(\overline{A} + \overline{B} + C)$$

Solución. Existen tres variables en el dominio y ocho posibles valores binarios, como se muestra en las tres columnas de la izquierda de la Tabla 4.7. Los valores binarios que hacen que los términos suma sean cero son $A + B + C$: 000; $A + \overline{B} + C$: 010; $A + \overline{B} + \overline{C}$: 011; $\overline{A} + B + \overline{C}$: 101; y $\overline{A} + \overline{B} + C$: 110. Para cada uno de estos valores binarios, se escribe un cero en la columna de salida como se indica en la tabla. Para las restantes combinaciones de valores binarios, se escribe un 1 en la columna de salida.

TABLA 4.7

Entradas			Salida
A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Obsérvese que la tabla de verdad de este ejemplo es la misma que la del Ejemplo 4.18. Esto significa que la suma de productos del ejemplo anterior y el producto de sumas de este ejemplo son equivalentes.

Problema relacionado. Desarrollar la tabla de verdad para el producto de sumas siguiente $(A + \overline{B} + C)\,(A + B + \overline{C})(\overline{A} + \overline{B} + \overline{C})$.

Determinación de las expresiones estándar a partir de una tabla de verdad

Para determinar la expresión de la suma de productos estándar representada por una tabla de verdad, se enumeran todos los valores de las variables de entrada para los que la salida es 1. Cada valor binario se convierte en el correspondiente término producto, reemplazando cada 1 por la variable y cada 0 por la variable complementada. Por ejemplo, el valor binario 1010 se transforma en un término producto de la manera siguiente:

$$1010 \longrightarrow A\overline{B}C\overline{D}$$

Para determinar el producto de sumas estándar representado por una tabla de verdad, se enumeran todos los valores binarios para los que la salida es 0. A continuación, se convierte cada valor binario en el correspondiente término suma, reemplazando cada 1 por la variable complementada, y cada 0 por la variable. Por ejemplo, el número binario 1001 se pasa a término suma de la manera siguiente:

$$1001 \longrightarrow \overline{A} + B + C + \overline{D}$$

EJEMPLO 4.20 A partir de la tabla de verdad de la Tabla 4.8, determinar la expresión suma de productos estándar y su expresión equivalente como producto de sumas estándar.

TABLA 4.8

Entradas			Salida
A	*B*	*C*	*X*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Solución. En la columna de salida hay cuatro 1s, y las combinaciones binarias correspondientes son 011, 100, 110 y 111. Estos valores se convierten a términos producto de la forma siguiente:

$$011 \longrightarrow \overline{A}BC$$
$$100 \longrightarrow A\overline{B}\,\overline{C}$$
$$110 \longrightarrow AB\overline{C}$$
$$111 \longrightarrow ABC$$

La suma de productos estándar resultante para la salida X es:

$$X = \overline{A}BC + A\overline{B}\,\overline{C} + AB\overline{C} + ABC$$

Para el producto de sumas, la salida es 0 para los valores 000, 001, 010 y 101. Estos valores binarios se transforman en términos suma de la forma siguiente:

$$000 \longrightarrow A + B + C$$
$$001 \longrightarrow A + B + \overline{C}$$
$$010 \longrightarrow A + \overline{B} + C$$
$$101 \longrightarrow \overline{A} + B + \overline{C}$$

El producto de sumas estándar que obtenemos a la salida X es:

$$X = (A + B + C)(A + B + \overline{C})(A + \overline{B} + C)(\overline{A} + B + \overline{C})$$

Problema relacionado. Sustituyendo los valores binarios, demostrar que las expresiones suma de productos y producto de sumas calculadas en el ejemplo anterior son equivalentes, es decir, para cada número binario que elijamos, ambas expresiones tienen que tomar el valor 0 o el valor 1, dependiendo del valor binario en cuestión.

REVISIÓN DE LA SECCIÓN 4.7

1. Si una determinada expresión booleana tiene un dominio de cinco variables, ¿cuántos valores binarios tendrá su tabla de verdad?
2. Si, en una tabla de verdad, la salida es 1 para el valor 0110, convertir este valor binario en su correspondiente operación producto, utilizando las variables *W, X, Y* y *Z*.
3. Si, en una tabla de verdad, la salida es 0 para el valor 1100, convertir este valor en su correspondiente operación suma, utilizando las variables *W, X, Y* y *Z*.

4.8 ■ MAPAS DE KARNAUGH

Un mapa de Karnaugh proporciona un método sistemático de simplificación de expresiones booleanas y, si se aplica adecuadamente, genera las expresiones suma de productos y producto de sumas más simples posibles. Como hemos visto, la efectividad de la simplificación algebraica depende de nuestra familiaridad con las leyes, reglas y teoremas del álgebra booleana y de nuestra habilidad a la hora de aplicarlas. Por otro lado, el mapa de Karnaugh es básicamente una "receta" para la simplificación.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Construir un mapa de Karnaugh de tres o cuatro variables. ■ Determinar el valor binario de cada celda de un mapa de Karnaugh. ■ Determinar el término producto estándar representado por cada celda de un mapa de Karnaugh. ■ Explicar la adyacencia de celdas e identificar celdas adyacentes.

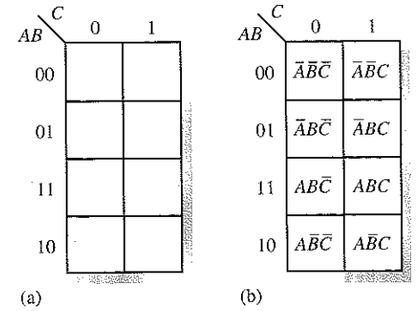
Un mapa de Karnaugh es similar a una tabla de verdad, ya que muestra todos los posibles valores de las variables de entrada y la salida resultante para cada valor. En vez de estar organizada en filas y columnas como una tabla de verdad, el mapa de Karnaugh es una secuencia de **celdas** en la que cada celda representa un valor binario de las variables de entrada. Las celdas se disponen de manera que la simplificación de una determinada expresión consiste en agrupar adecuadamente las celdas. Los mapas de Karnaugh pueden utilizarse para expresiones de dos, tres, cuatro y cinco variables, pero trataremos únicamente los casos de 3 y 4 variables para ilustrar su funcionamiento. La Sección 4.11 se ocupa de las expresiones de cinco variables que emplean un mapa de Karnaugh de 32 celdas. Existe otro método, que está fuera del propósito del libro, denominado método de Quine-McClusky, que puede usarse para un número de variables mayor.

El número de celdas de un mapa de Karnaugh es igual al número total de posibles combinaciones de las variables de entrada, al igual que el número de filas de una tabla de verdad. Para tres variables, el número de celdas necesarias es de $2^3 = 8$. Para cuatro variables, el número de celdas es de $2^4 = 16$.

Mapa de Karnaugh de tres variables

El mapa de Karnaugh de 3 variables es un conjunto de ocho celdas, como se muestra en la Figura 4.20(a). En este caso, *A, B* y *C* se utilizan para denominar a las variables, aunque se podrían usar otras letras. Los valores binarios de *A* y *B* se encuentran en la parte izquierda (observar la secuencia) y los valores de *C* en la parte superior.

FIGURA 4.20
Mapa de Karnaugh de tres variables.



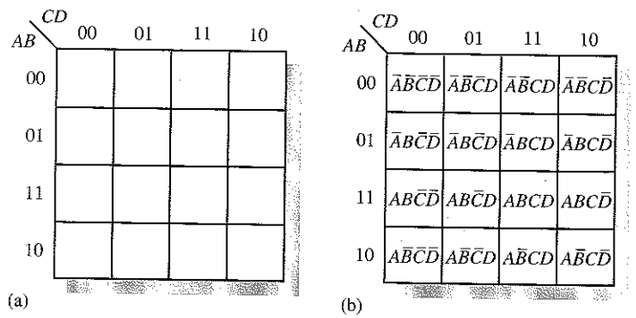
El valor de una determinada celda es el valor binario de A y B, en la parte izquierda de la misma fila, combinado con el valor de C en la parte superior de la misma columna. Por ejemplo, la celda de la esquina superior izquierda tiene un valor binario de 000 y la celda de la esquina inferior derecha tiene un valor de 101. La Figura 4.20(b) muestra los términos producto estándar representados por cada celda del mapa de Karnaugh.

Mapa de Karnaugh de cuatro variables

El mapa de Karnaugh de cuatro variables es una matriz de 16 celdas, como se muestra en la Figura 4.21(a). Los valores binarios de A y B se encuentran en la parte izquierda de la tabla, mientras que los de C y D están en la parte superior.

El valor de una determinada celda es el valor binario de A y B en la parte izquierda de la misma fila, combinado con los valores binarios de C y D en la parte superior de la misma columna. Por ejemplo, la celda de la esquina superior derecha tiene un valor de 0010 y la de la esquina inferior derecha tiene un valor de 1010. La Figura 4.21(b) muestra los términos producto estándar representados por cada celda del mapa de Karnaugh de cuatro variables.

FIGURA 4.21
Mapa de Karnaugh de cuatro variables.



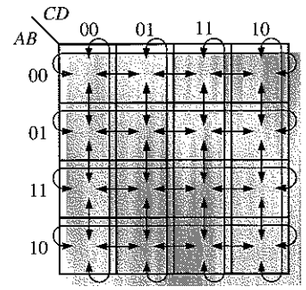
Adyacencia de celdas

Las celdas de un mapa de Karnaugh se disponen de manera que sólo cambia una única variable entre celdas adyacentes. La **adyacencia** se define por un cambio de una única variable. Las celdas que difieren en una única variable son adyacentes. Por ejemplo, en el mapa de 3 variables, la celda 010 es adyacente a la celda 000, a la 011 y a la 110. Las celdas cuyo valor difiere en más de una variable no son adyacentes. Por ejemplo, la celda 010 no es adyacente a la celda 001, a la 111, a la 100 ni a la 101.

Físicamente, cada celda es adyacente a las celdas que están situadas inmediatas a ella por cualquiera de sus cuatro lados. Una celda no es adyacente a aquellas que tocan diagonalmente alguna de sus esquinas. Además, las celdas de la fila superior son adyacentes a las de la fila inferior y las celdas de la columna izquierda son adyacentes a las situadas en la columna de la derecha. Esto se denomina adyacencia cíclica, ya que podemos pensar que el mapa de Karnaugh se dobla de forma que se toquen los extremos superior e inferior como si fuera un cilindro o los extremos de la derecha

e izquierda para formar la misma figura. La Figura 4.22 muestra la adyacencia de celdas de un mapa de Karnaugh de 4 variables, aunque las reglas de la adyacencia se aplican por igual a todas las tablas, independientemente de su número de variables.

FIGURA 4.22
Adyacencia de celdas en un mapa de Karnaugh. Las flechas apuntan a las celdas adyacentes.



REVISIÓN DE LA SECCIÓN 4.8

1. En un mapa de Karnaugh de tres variables, ¿cuál es el valor binario de cada una de las siguientes celdas?:
   (a) esquina superior izquierda (b) esquina inferior derecha
   (c) esquina inferior izquierda (d) esquina superior derecha
2. ¿Cuál es el producto estándar de cada celda de la cuestión 1 para las variables X, Y y Z?
3. Repetir la cuestión 1 para un mapa de cuatro variables.
4. Repetir la cuestión 2 para un mapa de cuatro variables utilizando las variables W, X, Y y Z.

4.9 ■ MINIMIZACIÓN DE UNA SUMA DE PRODUCTOS MEDIANTE EL MAPA DE KARNAUGH

Como se ha enunciado en la sección anterior, el mapa de Karnaugh se utiliza para reducir expresiones booleanas a su mínima expresión. Una expresión suma de productos minimizada está formada por el mínimo número de términos producto posibles con el mínimo número de variables por término. Generalmente, una expresión suma de productos minimizada puede ser implementada mediante un número de puertas menor que su expresión estándar, lo cual constituye la finalidad del proceso de simplificación.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Representar una expresión suma de productos en un mapa de Karnaugh. ■ Combinar los 1s del mapa en grupos máximos. ■ Determinar los términos producto mínimo de cada grupo del mapa. ■ Combinar los términos producto mínimo para formar una expresión suma de productos mínima. ■ Convertir una tabla de verdad en un mapa de Karnaugh para simplificar la correspondiente expresión. ■ Utilizar condiciones "indiferentes" en un mapa de Karnaugh.

Mapa de Karnaugh de una suma de productos estándar

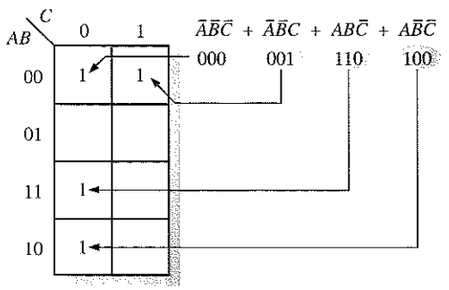
Por cada término de la expresión suma de productos, se coloca un 1 en el mapa de Karnaugh en la celda correspondiente al valor del producto. Por ejemplo, para el término $A\overline{B}C$ se situaría un 1 en la celda 101 en un mapa de Karnaugh de 3 variables.

Cuando se ha completado el mapa de Karnaugh correspondiente a la suma de productos dada, en dicho mapa habrá tantos unos como términos en la expresión. Las celdas que no tienen un 1 son aquellas para las que la expresión es 0. Generalmente, cuando se trabaja con una expresión suma de productos, los 0s se dejan fuera del mapa. Los siguientes pasos y la Figura 4.23 nos ilustran cómo completar los mapas de Karnaugh.

Paso 1. Determinar el valor binario de cada término producto de la suma de productos estándar. Tras un poco de práctica, la evaluación de términos la podremos realizar mentalmente.

Paso 2. A medida que evaluamos cada término, colocamos un 1 en el mapa de Karnaugh, en la celda que tiene el mismo valor que dicho término.

FIGURA 4.23
Ejemplo de transformación a mapa de Karnaugh de una suma de productos estándar.



En los dos ejemplos siguientes se verá este proceso de transformación.

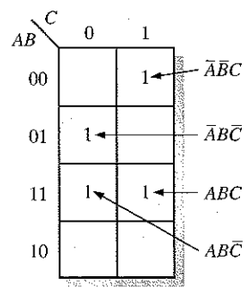
EJEMPLO 4.21 Transformar la siguiente suma de productos estándar en un mapa de Karnaugh:

$$\overline{A}\,\overline{B}C + \overline{A}B\overline{C} + AB\overline{C} + ABC$$

Solución. La expresión se evalúa como se muestra más abajo. Se introduce un 1 en el mapa de Karnaugh de tres variables de la Figura 4.24 por cada producto estándar de la expresión.

$$\overline{A}\,\overline{B}C + \overline{A}B\overline{C} + AB\overline{C} + ABC$$
$$001 \quad 010 \quad 110 \quad 111$$

FIGURA 4.24



Problema relacionado. Transformar la expresión estándar de la suma de productos $ABC + A\overline{B}C + ABC$ en un mapa de Karnaugh.

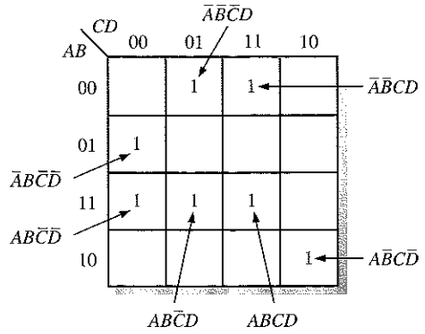
EJEMPLO 4.22 Transformar la siguiente suma de productos estándar en un mapa de Karnaugh:

$$\overline{A}\,\overline{B}CD + \overline{A}B\overline{C}\,\overline{D} + AB\overline{C}D + ABCD + AB\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + A\overline{B}\overline{C}\overline{D}$$

Solución. La expresión se evalúa como se muestra a continuación. Se coloca un 1 en el mapa de Karnaugh de la Figura 4.25 por cada producto estándar de la expresión.

$$\overline{A}\,\overline{B}CD + \overline{A}B\overline{C}\,\overline{D} + AB\overline{C}D + ABCD + AB\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + A\overline{B}\overline{C}\overline{D}$$
$$0011 \quad 0100 \quad 1101 \quad 1111 \quad 1100 \quad 0001 \quad 1010$$

FIGURA 4.25



Problema relacionado. Transformar la siguiente expresión estándar suma de productos en un mapa de Karnaugh.

$$\overline{A}BC\overline{D} + ABC\overline{D} + AB\overline{C}\,\overline{D} + ABCD$$

Mapa de Karnaugh de una suma de productos no estándar

Antes de poder utilizar un mapa de Karnaugh, las expresiones booleanas deben estar en su forma estándar. Si una expresión no lo está, se pasará al formato estándar mediante el procedimiento descrito en la Sección 4.6 o mediante desarrollo numérico. Dado que, en cualquier caso, las expresiones tienen que evaluarse antes de pasarlas al mapa de Karnaugh, el desarrollo numérico es quizá el método más eficaz.

Desarrollo numérico de un producto no estándar. Recuerde que a un término en forma no estándar le faltan una o más variables en su expresión. Por ejemplo, supongamos que uno de los productos de una cierta suma de productos de tres variables es $A\overline{B}$. Este término se puede desarrollar numéricamente para obtener una expresión estándar de la manera siguiente. En primer lugar, se escribe el valor binario de las dos variables

y le añadimos un 0 que corresponde a la variable que falta, $\overline{C}$:100. A continuación, se escribe el valor binario de las dos variables y le añadimos un 1, que corresponde a la variable que falta, C:101. Los dos valores binarios resultantes son los valores respectivos de los términos de la expresión suma de productos estándar $A\overline{B}\,\overline{C}$ y $A\overline{B}C$.

Veamos otro ejemplo; se supone que uno de los términos de una expresión de tres variables es B (recuérdese que una variable única se considera como un término producto en una suma de productos). Este término se puede desarrollar numéricamente para obtener su formato estándar del siguiente modo: se escribe el valor binario de la variable y luego se añaden todos los posibles valores de las variables que faltan, A y C:

B
010
011
110
111

Los cuatro números binarios resultantes son los valores correspondientes a la suma de productos estándar, es decir, $\overline{A}B\overline{C}$, $\overline{A}BC$, $AB\overline{C}$ y ABC.

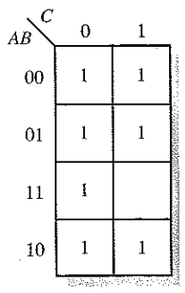
EJEMPLO 4.23 Transformar la siguiente expresión suma de productos en un mapa de Karnaugh: $\overline{A} + A\overline{B} + AB\overline{C}$.

Solución. Obviamente, la suma de productos no está en formato estándar, ya que cada término no contiene las tres variables. El primer término no posee dos de las tres variables; el segundo carece de una, mientras que el tercero sí que es estándar. En primer lugar, desarrollamos los términos numéricamente de la forma que sigue:

$\overline{A}$	$+A\overline{B}$	$+AB\overline{C}$
000	100	110
001	101	
010		
011		

Cada uno de los valores binarios resultantes se traslada al mapa, colocando un 1 en la celda apropiada del mapa de Karnaugh de 3 variables de la Figura 4.26.

FIGURA 4.26



Problema relacionado. Transformar la expresión suma de productos $BC + \overline{A}\,\overline{C}$ en un mapa de Karnaugh.

EJEMPLO 4.24 Transformar la siguiente expresión suma de productos en un mapa de Karnaugh:

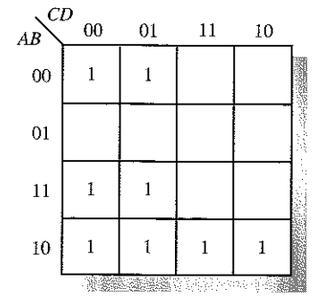
$$\overline{B}\,\overline{C} + A\overline{B} + AB\overline{C} + A\overline{B}C\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + A\overline{B}CD$$

Solución. Obviamente, la suma de productos no está escrita en forma estándar, ya que cada término no contiene las cuatro variables. A los dos primeros les faltan dos variables, mientras que los dos restantes sí son estándar. En primer lugar, se desarrollan numéricamente los términos como sigue:

$\overline{B}\,\overline{C}$	$+A\overline{B}$	$+AB\overline{C}$	$+A\overline{B}C\overline{D}$	$+\overline{A}\,\overline{B}\,\overline{C}D$	$+A\overline{B}CD$
0000	1000	1100	1010	0001	1011
0001	1001	1101			
1000	1010				
1001	1011				

Cada uno de los valores binarios resultantes se traslada al mapa, situando un 1 en la celda correspondiente del mapa de Karnaugh de 4 variables de la Figura 4.27. Nótese que algunos de los valores de la expresión desarrollada son redundantes.

FIGURA 4.27



Problema relacionado. Transformar la expresión $A + \overline{C}D + AC\overline{D} + \overline{A}BC\overline{D}$ en un mapa de Karnaugh.

Simplificación de una suma de productos mediante el mapa de Karnaugh

El proceso que genera una expresión que contiene el menor número posible de términos con el mínimo número de variables posibles se denomina **minimización**. Después de haber obtenido el mapa de Karnaugh de una suma de productos, se deben seguir tres pasos para obtener la expresión suma de productos mínima: agrupar los 1s, determinar el término producto correspondiente a cada grupo y sumar los términos producto obtenidos.

Agrupación de 1s. Podemos agrupar los 1s del mapa de Karnaugh de acuerdo con las reglas siguientes, rodeando las celdas adyacentes que contengan 1s. La finalidad es maximizar el tamaño de los grupos y minimizar el número de estos grupos.

1. Un grupo tiene que contener 1, 2, 4, 8 o 16 celdas. En el caso de un mapa de Karnaugh de 3 variables, el grupo máximo puede contener 8 celdas.
2. Cada celda de un grupo tiene que ser adyacente a una o más celdas del mismo grupo, pero no todas las celdas del grupo tienen que ser adyacentes entre sí.
3. Incluir siempre en cada grupo el mayor número posible de 1s de acuerdo a la regla número 1.
4. Cada 1 del mapa tiene que estar incluido en al menos un grupo. Los 1s que ya pertenezcan a un grupo pueden estar incluidos en otro, siempre que los grupos que se solapen contengan 1s no comunes.

EJEMPLO 4.25 Agrupar los 1s en cada uno de los mapas de Karnaugh de la Figura 4.28.

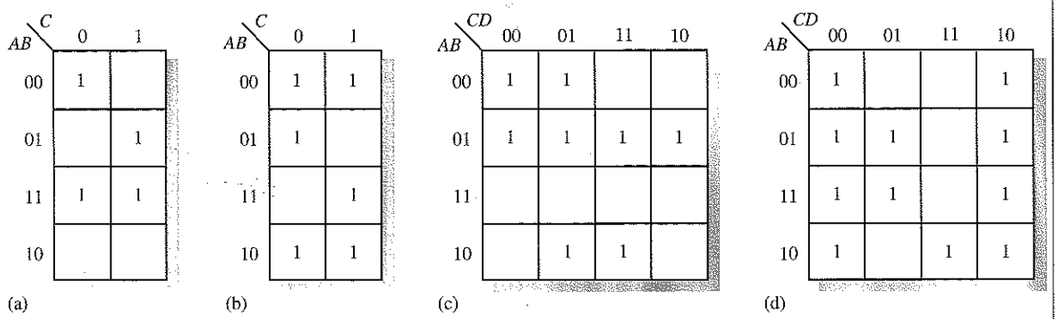


FIGURA 4.28

Solución. En la Figura 4.29 se muestran los grupos. En algunos casos, puede existir más de una forma de agrupar los 1s para formar grupos máximos.

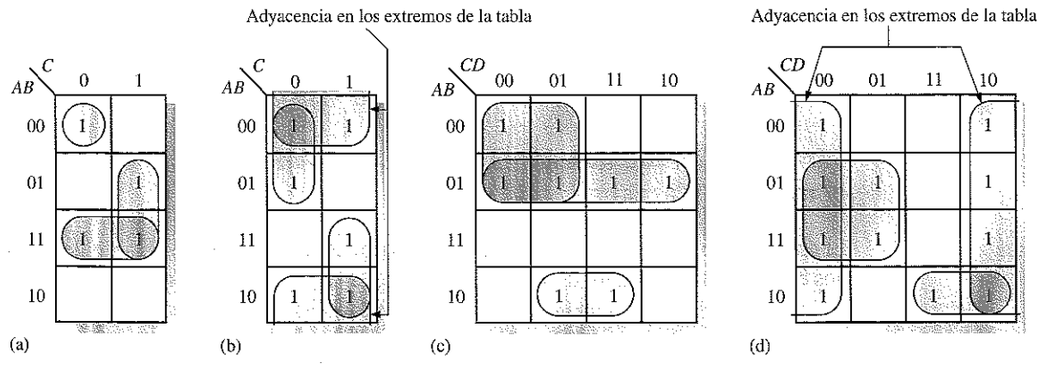


FIGURA 4.29

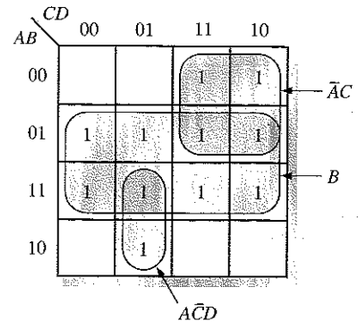
Problema relacionado. Determinar si existen otras formas de agrupar los 1s en la Figura 4.29, para obtener un número mínimo de grupos máximos.

Determinación de la expresión suma de productos mínima a partir del mapa. Cuando todos los 1s que representan los términos productos estándar de una expresión se han trasladado al mapa y se han agrupado adecuadamente, comienza el proceso de obtención de la suma de productos mínima. Para encontrar los términos mínimos y la expresión suma de productos mínima se aplican las siguientes reglas:

1. Agrupar las celdas que contienen 1s. Cada grupo de celdas que contiene 1s da lugar a un término producto compuesto por todas las variables que aparecen en el grupo en sólo una forma (no complementada o complementada). Las variables que aparecen complementadas y sin complementar dentro del mismo grupo se eliminan. A éstas se les denomina *variables contradictorias*.
2. Determinar la operación producto mínima para cada grupo.
   (a) Para un mapa de 3 variables:
      (1) Un grupo formado por una única celda da lugar a un término producto de tres variables.
      (2) Un grupo formado por 2 celdas da lugar a un término producto de dos variables.
      (3) Un grupo formado por 4 celdas da lugar a un término de una variable.
      (4) Un grupo formado por 8 celdas indica que la expresión vale 1.
   (b) Para un mapa de 4 variables:
      (1) Un grupo formado por 1 celda da lugar a un término producto de cuatro variables.
      (2) Un grupo formado por 2 celdas da lugar a un término producto de tres variables.
      (3) Un grupo formado por 4 celdas da lugar a un término producto de dos variables.
      (4) Un grupo formado por 8 celdas da lugar a un término de una variable.
      (5) Un grupo formado por 16 celdas indica que la expresión vale 1.
3. Cuando se han obtenido todos los términos mínimos a partir del mapa de Karnaugh, se suman para obtener la expresión suma de productos mínima.

EJEMPLO 4.26 Determinar los productos para el mapa de Karnaugh de la Figura 4.30 y escribir la expresión suma de productos mínima resultante.

FIGURA 4.30



Solución. En la Figura 4.30, el producto correspondiente al grupo de 8 celdas es B, puesto que las celdas que forman el grupo contienen A y $\overline{A}$, C y $\overline{C}$, y D y $\overline{D}$, por lo que estas variables se eliminan. En el grupo formado por 4 celdas, aparecen B, $\overline{B}$, D, y $\overline{D}$, de forma que el producto es $\overline{A}C$. El grupo formado por 2 celdas, contiene a B y $\overline{B}$, siendo $A\overline{C}D$ el producto. Obsérvese cómo se superponen los grupos para que éstos sean de tamaño máximo. La expresión suma de productos mínima que resulta es la suma de todos los productos obtenidos.

$$B + \overline{A}C + A\overline{C}D$$

Problema relacionado. En el mapa de Karnaugh de la Figura 4.30, añadir un 1 a la celda inferior derecha (1010) y determinar la expresión suma de productos resultante.

EJEMPLO 4.27 Determinar los productos para cada uno de los mapas de Karnaugh de la Figura 4.31 y escribir las correspondientes expresiones suma de productos mínima resultante.

Solución. En la Figura 4.31 se muestran los productos mínimos resultantes para cada grupo. La expresión suma de productos mínima para cada uno de los mapas de Karnaugh de la figura son:

(a) $AB + BC + \overline{A}\,\overline{B}\,\overline{C}$ **(b)** $\overline{B} + \overline{A}\,\overline{C} + AC$
(c) $\overline{A}B + \overline{A}\,\overline{C} + A\overline{B}D$ **(d)** $\overline{D} + A\overline{B}C + B\overline{C}$

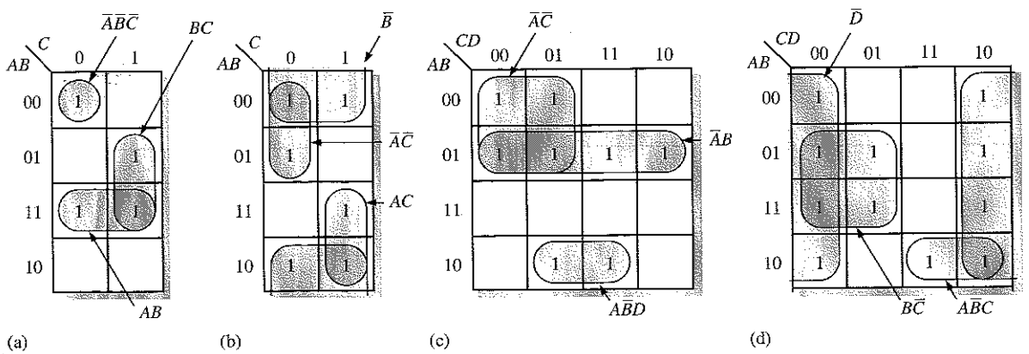


FIGURA 4.31

Problema relacionado. En el mapa de Karnaugh de la Figura 4.31(d), añadir un 1 a la celda 0111 y determinar la expresión suma de productos resultante.

EJEMPLO 4.28 Mediante el mapa de Karnaugh minimizar la suma de productos estándar siguiente:

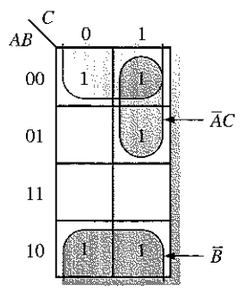
$$A\overline{B}C + \overline{A}BC + \overline{A}\,\overline{B}C + \overline{A}\,\overline{B}\,\overline{C} + A\overline{B}\,\overline{C}$$

Solución. Los valores binarios de la expresión son:

$$101 + 011 + 011 + 000 + 100$$

La suma de productos estándar se pasa al mapa y las celdas se agrupan como se muestra en la Figura 4.32.

FIGURA 4.32



Obsérvese el grupo de 4 celdas en los extremos del mapa que incluye las filas superior e inferior de 1s. El 1 restante se incluye en otro grupo superpuesto de dos celdas. El grupo de los cuatro 1s origina un término con una única variable, $\overline{B}$. Esto se deduce del hecho de que, dentro del grupo, $\overline{B}$ es la única variable que no varía de celda a celda. El grupo de los dos 1s da lugar a un producto de dos variables $\overline{A}$C. Éste término se obtiene observando que, dentro de este grupo, $\overline{A}$ y C no varían de una celda a la siguiente. La correspondiente expresión suma de productos mínima es:

$$\overline{B} + \overline{A}C$$

No debe olvidarse que esta expresión mínima es equivalente a la expresión estándar original.

Problema relacionado. Utilizando un mapa de Karnaugh, simplificar la siguiente expresión suma de productos estándar:

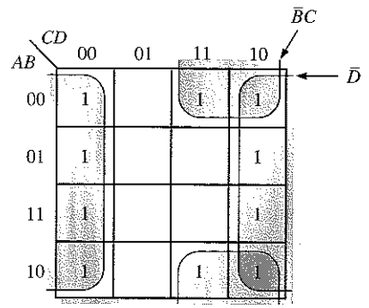
$$X\overline{Y}Z + XY\overline{Z} + \overline{X}YZ + \overline{X}Y\overline{Z} + X\overline{Y}\,\overline{Z} + XYZ$$

EJEMPLO 4.29 Mediante un mapa de Karnaugh, minimizar la expresión suma de productos siguiente:

$$\overline{B}\,\overline{C}D + \overline{A}B\overline{C}D + AB\overline{C}\,\overline{D} + \overline{A}\,\overline{B}CD + A\overline{B}CD + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}BC\overline{D} + ABC\overline{D} + A\overline{B}\,\overline{C}\,\overline{D}$$

Solución. El primer término $\overline{B}\,\overline{C}\,\overline{D}$ tiene que desarrollarse para obtener una suma de productos estándar, que será $A\overline{B}\,\overline{C}\,\overline{D}$ y $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$, que se traslada al mapa, y las celdas se agrupan como se muestra en la Figura 4.33.

FIGURA 4.33



Obsérvese que ambos grupos tienen adyacencia "cíclica" de celdas. Se puede formar el grupo de ocho celdas, ya que las dos columnas exteriores son adyacentes. El grupo de cuatro celdas se forma tomando los dos restantes 1s, puesto que las celdas superior e inferior son adyacentes. Se indica el término producto para cada grupo y la expresión suma de productos mínima resultante es:

$$\overline{D} + \overline{B}C$$

No se debe olvidar que esta expresión mínima es equivalente a la expresión estándar original.

Problema relacionado. Mediante un mapa de Karnaugh simplificar la siguiente expresión suma de productos:

$$\overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z} + W\overline{X}YZ + W\overline{X}\,\overline{Y}Z + \overline{W}YZ + W\overline{X}\,\overline{Y}\,\overline{Z}$$

Obtención directa del mapa de Karnaugh a partir de la tabla de verdad

Hemos visto cómo las expresiones booleanas se transforman en mapas de Karnaugh. Ahora se aprenderá cómo pasar de una tabla de verdad a un mapa de Karnaugh. Recuérdese que una tabla de verdad proporciona la salida de una expresión booleana para todas las posibles combinaciones de las variables de entrada. En la Figura 4.34 se facilita un ejemplo de expresión booleana junto con su tabla de verdad. Obsérvese que la salida X es 1 para cuatro distintas combinaciones de las variables de entrada. Los unos de la columna de salida de la tabla de verdad se trasladan directamente al mapa de Karnaugh, a las celdas correspondientes a los valores asociados de las combinaciones de variables de entrada, como muestra la Figura 4.34. En esta figura puede verse que tanto la expresión booleana, la tabla de verdad como el mapa de Karnaugh son sólo distintas maneras de representar una función lógica.

Condiciones indiferentes

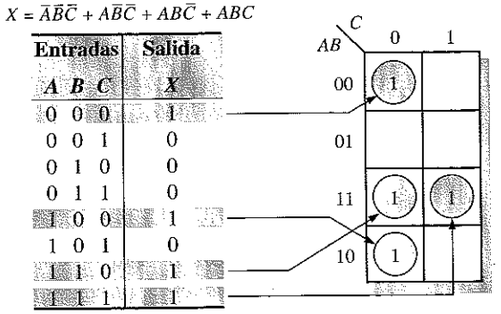
Algunas veces se producen situaciones en las que algunas combinaciones de las variables de entrada no están permitidas. Por ejemplo, recuérdese que en el código BCD, visto en el Capítulo 2, existían seis combinaciones no válidas: 1010, 1011, 1100, 1101,

1110 y 1111. Dado que estos estados no permitidos no ocurren nunca en una aplicación que emplee el código BCD, pueden considerarse como términos **indiferentes** con respecto a su efecto en la salida. Esto significa que a estos términos se les puede asignar tanto un 1 como un 0 en la salida; realmente no son importantes dado que nunca van a generarse.

FIGURA 4.34
Ejemplo de obtención directa de un mapa de Karnaugh a partir de una tabla de verdad.

$X = \overline{A}\,\overline{B}\,\overline{C} + A\overline{B}\,\overline{C} + AB\overline{C} + ABC$

Entradas			Salida
A	*B*	*C*	*X*
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1



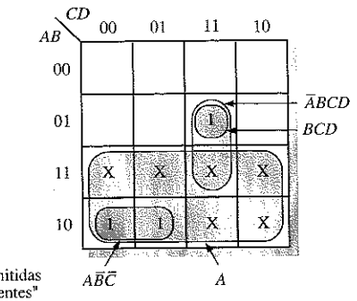
Los términos "indiferentes" pueden utilizarse para aprovechar mejor el método del mapa de Karnaugh. La Figura 4.35 muestra que, para cada término indiferente, se escribe una X en la celda. Cuando se agrupan los 1s, las X pueden ser consideradas también como 1s para agrandar los grupos, o como 0s si no obtenemos ninguna ventaja. Cuanto mayor sea el grupo, más sencillo será el término resultante.

La tabla de verdad de la Figura 4.35(a) describe una función lógica que tiene sólo salida igual a 1 cuando el código BCD de 7, 8 o 9 se introduce a la entrada. Utilizando los términos indiferentes como unos, el resultado que obtenemos para la función es $A + BCD$. Si no se emplean estos términos "indiferentes", la expresión obtenida sería $A\,\overline{B}\,\overline{C} + \overline{A}\,BCD$. Luego puede verse que el utilizar los términos indiferentes contribuye a obtener una expresión más sencilla.

FIGURA 4.35
Ejemplo de la utilización de las condiciones indiferentes para simplificar una expresión.

Entradas				Salida
A	*B*	*C*	*D*	*Y*
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

(a) Tabla de verdad



(b) Sin condiciones "indiferentes" $Y = A\overline{B}\,\overline{C} + \overline{A}BCD$
Con condiciones "indiferentes" $Y = A + BCD$

REVISIÓN DE LA SECCIÓN 4.9

1. Explicar los mapas de Karnaugh de 3 y 4 variables.
2. Agrupar los 1s y escribir la expresión suma de productos simplificada para el mapa de Karnaugh de la Figura 4.24.
3. Escribir la expresión estándar de la suma de productos original de cada uno de los mapas de Karnaugh de la Figura 4.31.

4.10 ■ MINIMIZACIÓN DE UN PRODUCTO DE SUMAS MEDIANTE EL MAPA DE KARNAUGH

En la sección anterior estudiamos la minimización de una expresión suma de productos mediante los mapas de Karnaugh. En esta sección, nos vamos a centrar en las expresiones producto de sumas. Los métodos son muy similares, excepto que ahora se trata de productos de sumas, en los que los 0s representan las operaciones de suma estándar y se colocan en el mapa de Karnaugh en lugar de los 1s. Esta sección se puede omitir sin que ello afecte al resto de los temas que se van a tratar.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Transformar un producto de sumas estándar en un mapa de Karnaugh. ■ Combinar los 0s del mapa para formar grupos máximos. ■ Determinar el número mínimo de términos para cada grupo del mapa. ■ Combinar los términos suma mínimos para formar el producto de sumas mínimo. ■ Utilizar el mapa de Karnaugh para convertir productos de sumas en sumas de productos, y viceversa.

Conversión de una expresión producto de sumas estándar a mapa de Karnaugh

Para un producto de sumas en forma estándar, se introduce un 0 en el mapa de Karnaugh por cada término suma de la expresión. Cada 0 se sitúa en la celda correspondiente al valor de un término suma. Por ejemplo, para la suma $A + \overline{B} + C$, se escribe un 0 en la celda 010 del mapa de Karnaugh de 3 variables.

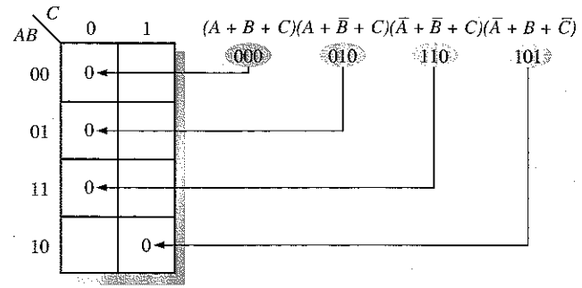
Cuando un producto de sumas se ha trasladado por completo al mapa, habrá tantos 0s en el mapa como términos suma en la expresión del producto de sumas estándar. Las celdas que no contienen un 0 son aquellas para las que la expresión vale 1. Generalmente, cuando se trabaja con productos de sumas, los 1s no se escriben. Los siguientes pasos junto con la Figura 4.36 ilustran este proceso.

Paso 1. Determinar el valor binario de cada término suma del producto de sumas estándar. Éste es el valor binario que hace que dicho término sea igual a 0.

Paso 2. Cada vez que se evalúa un término suma, se introduce un 0 en la correspondiente celda del mapa de Karnaugh.

El Ejemplo 4.30 nos ilustrará más detalladamente este proceso.

FIGURA 4.36
Ejemplo de obtención del mapa de Karnaugh de un producto de sumas estándar.



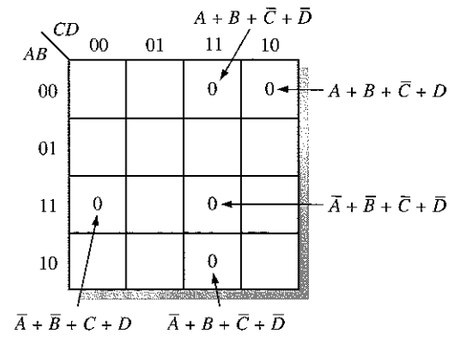
EJEMPLO 4.30 Transformar la siguiente expresión suma de productos estándar en un mapa de Karnaugh:

$$(\overline{A} + \overline{B} + C + D)(\overline{A} + B + \overline{C} + \overline{D})(A + B + \overline{C} + D)(\overline{A} + \overline{B} + \overline{C} + \overline{D})(A + B + \overline{C} + \overline{D})$$

Solución. La expresión se evalúa como se indica más abajo y se coloca un 0 en el mapa de Karnaugh de la Figura 4.37 por cada término suma estándar de la expresión.

$$\underset{1100}{(\overline{A} + \overline{B} + C + D)}\underset{1011}{(\overline{A} + B + \overline{C} + \overline{D})}\underset{0010}{(A + B + \overline{C} + D)}\underset{1111}{(\overline{A} + \overline{B} + \overline{C} + \overline{D})}\underset{0011}{(A + B + \overline{C} + \overline{D})}$$

FIGURA 4.37



Problema relacionado. Transformar la siguiente expresión suma de productos estándar en un mapa de Karnaugh.

$$(A + \overline{B} + \overline{C} + D)(A + B + C + \overline{D})(A + B + C + D)(\overline{A} + B + \overline{C} + D)$$

Simplificación mediante el mapa de Karnaugh de expresiones producto de sumas

El proceso de minimización de un producto de sumas es básicamente el mismo que para una expresión suma de productos, excepto que ahora hay que agrupar los 0s para generar el mínimo número de términos suma, en lugar de los 1s para obtener el núme-

ro mínimo de términos producto. Las reglas para agrupar los 0s son las mismas que para agrupar los 1s, y son las que se han estudiado en la Sección 4.9.

EJEMPLO 4.31 Utilizar el mapa de Karnaugh para minimizar el siguiente producto de sumas estándar:

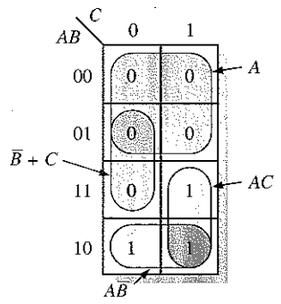
$$(A + B + C)(A + B + \overline{C})(A + \overline{B} + C)(A + \overline{B} + \overline{C})(\overline{A} + \overline{B} + C)$$

Solución. Las combinaciones de valores binarios de la expresión son:

$$(0 + 0 + 0)(0 + 0 + 1)(0 + 1 + 0)(0 + 1 + 1)(1 + 1 + 0)$$

La expresión de la suma de productos estándar se traslada al mapa de Karnaugh, y las celdas se agrupan como se muestra en la Figura 4.38

FIGURA 4.38



Obsérvese que el 0 de la celda 110 se incluye en un grupo de dos celdas, utilizando el 0 del grupo de cuatro celdas. El término suma para cada grupo se muestra en la figura y la expresión suma de productos mínima resultante es:

$$A(\overline{B} + C)$$

Téngase en cuenta que esta expresión suma de productos mínima es equivalente a la expresión suma de productos estándar.

Agrupando los 1s como se indica en las áreas de color gris se obtiene una expresión suma de productos que es equivalente a agrupar los 0s.

$$AC + A\overline{B} = A(\overline{B} + C)$$

Problema relacionado. Utilizar un mapa de Karnaugh para simplificar la siguiente suma de productos estándar:

$$(X + \overline{Y} + Z)(X + \overline{Y} + \overline{Z})(\overline{X} + \overline{Y} + Z)(\overline{X} + Y + Z)$$

EJEMPLO 4.32 Utilizar un mapa de Karnaugh para minimizar el siguiente producto de sumas:

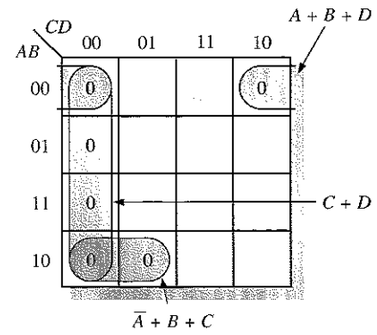
$$(B + C + D)(A + B + \overline{C} + D)(\overline{A} + B + C + \overline{D})(A + \overline{B} + C + D)(\overline{A} + \overline{B} + C + D)$$

Solución. El primer término tiene que ampliarse a $\overline{A} + B + C + D$ y $A + B + C + D$ para conseguir una expresión producto de sumas estándar, que luego se pasa a un mapa de Karnaugh, y en la que las celdas se agrupan como muestra la Figura 4.39. En ella se indican los términos suma correspondientes a cada grupo, y el producto de sumas mínimo es:

$$(C + D)(A + B + D)(\overline{A} + B + C)$$

Recuerde que esta expresión mínima es equivalente al producto de sumas original.

FIGURA 4.39



Problema relacionado. Utilizar un mapa de Karnaugh para simplificar la siguiente expresión producto de sumas:

$$(W + \overline{X} + Y + \overline{Z})(W + X + Y + Z)(W + \overline{X} + \overline{Y} + Z)(\overline{W} + \overline{X} + Z)$$

Conversión entre suma de productos y productos de sumas mediante el mapa de Karnaugh

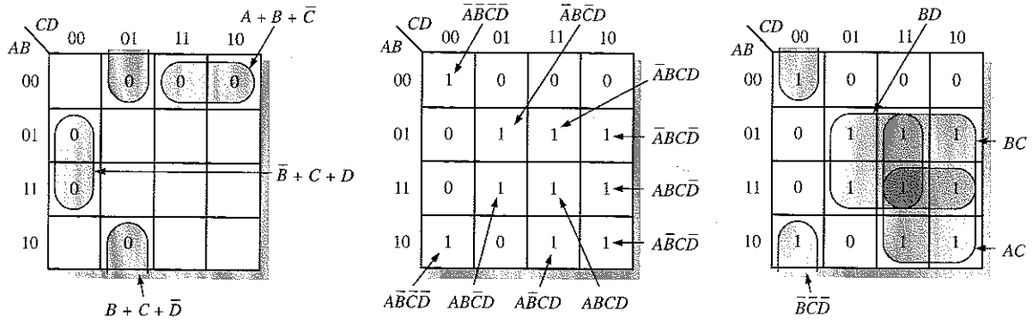
Cuando un producto de sumas se traslada a un mapa de Karnaugh, puede fácilmente pasarse a la suma de productos equivalente directamente a partir de dicho mapa. También, dado un mapa de Karnaugh de una suma de productos, el producto de sumas equivalente puede obtenerse directamente a partir del mapa. Esto proporciona una excelente manera de comparar ambas formas mínimas de una expresión, para determinar si una de ellas puede implementarse con menos puertas que la otra.

Para un producto de sumas, todas las celdas que no contienen 0s contienen 1s, de lo que se deriva su expresión suma de productos. De igual manera, para una suma de productos, todas las celdas que no contienen 1s contendrán 0s, de los que se obtiene la expresión producto de sumas. El Ejemplo 4.33 ilustra esta conversión.

EJEMPLO 4.33 Utilizando un mapa de Karnaugh, convertir el siguiente producto de sumas estándar en: un producto de sumas mínimo, una suma de productos estándar y una suma de productos mínima.

$$(\overline{A} + \overline{B} + C + D)(A + \overline{B} + C + D)(A + B + C + \overline{D})$$
$$(A + B + \overline{C} + \overline{D})(\overline{A} + B + C + \overline{D})(A + B + \overline{C} + D)$$

Solución. Los ceros de la expresión producto de sumas estándar se transforman y agrupan para obtener el producto de sumas mínimo, como se indica en la Figura 4.40(a). En la Figura 4.40(b), se añaden 1s en las celdas que no contienen 0s. De cada celda que contenga un 1, se obtiene un término producto estándar, como se indica. Estos términos producto forman la expresión suma de productos estándar. En la Figura 4.40(c), se agrupan los 1s y se obtiene una expresión suma de productos mínima.



(a) Producto de sumas mínimo:
$(A + B + \overline{C})(\overline{B} + C + D)(B + C + \overline{D})$

(b) Suma de productos estándar:
$\overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}\overline{C}D + \overline{A}B\overline{C}D + \overline{A}BC\overline{D} + AB\overline{C}D +$
$\overline{A}BCD + A\overline{B}\overline{C}\overline{D} + A\overline{B}CD + AB\overline{C}D + A\overline{B}CD + ABCD$

(c) Suma de productos mínima:
$AC + BC + BD + \overline{B}\overline{C}\overline{D}$

FIGURA 4.40

Problema relacionado. Utilizar un mapa de Karnaugh para convertir la siguiente expresión a su forma suma de productos mínima:

$$(W + \overline{X} + Y + \overline{Z})(\overline{W} + X + \overline{Y} + \overline{Z})(\overline{W} + \overline{X} + \overline{Y} + Z)(\overline{W} + \overline{X} + \overline{Z})$$

REVISIÓN DE LA SECCIÓN 4.10

1. ¿Cuál es la diferencia entre pasar a mapa de Karnaugh un producto de sumas y una suma de productos?
2. ¿Cuál es el término suma estándar para un 0 en la celda 1011?
3. ¿Cuál es el término producto estándar para un 1 en la celda 0010?

4.11 ■ MAPA DE KARNAUGH DE CINCO VARIABLES

Las funciones booleanas de cinco variables pueden simplificarse mediante un mapa de Karnaugh de 32 celdas. Realmente, para construir un mapa de 5 variables se utilizan dos mapas de cuatro variables (con 16 celdas cada uno). Ya conocemos la adyacencia de celdas en los mapas de 4 variables y cómo se forman los grupos de celdas que contengan 1s para simplificar una suma de productos. Luego todo lo que se necesita aprender para manejar cinco variables es la adyacencia de celdas entre dos mapas de 4 variables y cómo agruparlas. Esta sección puede ser omitida sin afectar al resto de las materias que se van a tratar más adelante.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Determinar la adyacencia de celdas en un mapa de 5 variables. ■ Formar grupos que contengan el máximo número de celdas en un mapa de 5 variables. ■ Minimizar las expresiones booleanas de 5 variables utilizando el mapa de Karnaugh.

Un mapa de Karnaugh de cinco variables (*ABCDE*) puede crearse utilizando dos mapas de 4 variables, con los que ya estamos familiarizados. Cada mapa contiene 16 celdas con todas las posibles combinaciones de las variables *B, C, D* y *E*. Un mapa es para $A = 0$, mientras que el otro es para $A = 1$, como se muestra en la Figura 4.41.

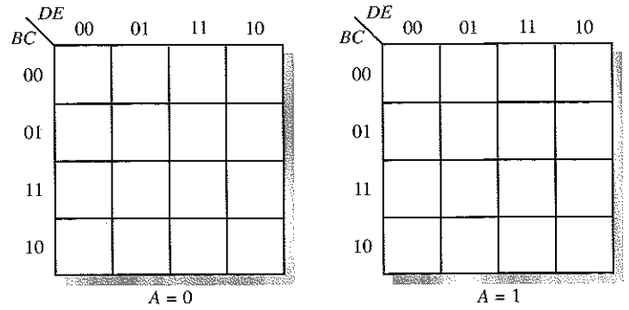


FIGURA 4.41
Mapa de Karnaugh de 5 variables.

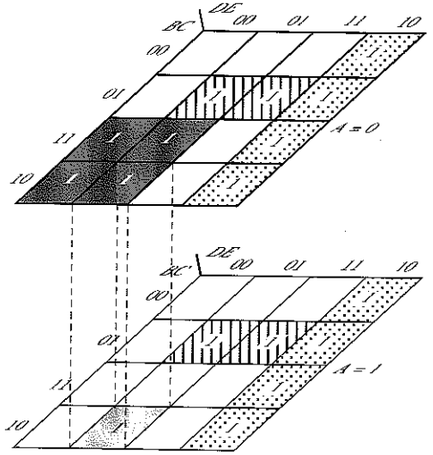
Adyacencia de celdas

Ya conocemos cómo determinar celdas adyacentes dentro de un mapa de cuatro variables. La mejor manera de visualizar la adyacencia de celdas entre los dos mapas de 16 celdas consiste en imaginar que el mapa $A = 0$ está colocado encima del mapa $A = 1$. Cada celda del mapa $A = 0$ es adyacente con la celda que está justo debajo en el mapa $A = 1$.

Para ilustrar esto, la Figura 4.42 muestra un ejemplo con cuatro grupos, con los mapas en disposición tridimensional. Los 1s de las celdas punteadas forman un grupo de 8 bits (cuatro correspondientes al mapa $A = 0$ combinadas con cuatro del mapa $A = 1$). Los 1s de las celdas rayadas forman un grupo de 4 bits. Los unos de las celdas

gris oscuro constituyen un grupo de 4 bits sólo en el mapa $A = 0$. El 1 de la celda gris claro del mapa $A = 1$ se agrupa con el 1 de la celda gris oscuro de la parte inferior derecha del mapa $A = 0$ para formar un grupo de 2 bits.

FIGURA 4.42
Ilustración de la agrupación de 1s en celdas adyacentes de un mapa de 5 variables.



Expresión booleana simplificada

La expresión booleana suma de productos original que está dibujada en el mapa de Karnaugh de la Figura 4.42 contiene 17 términos de cinco variables, ya que existen diecisiete 1s en el mapa. Como ya sabemos, sólo las variables que no cambian dentro de un grupo permanecen en la expresión correspondiente a ese grupo. La expresión simplificada se obtiene a partir del mapa de la manera siguiente:

- El término del grupo punteado es $D\overline{E}$.
- El término del grupo rayado es $\overline{B}CE$.
- El término del grupo gris oscuro es $\overline{A}B\overline{D}$.
- El término de la celda gris claro junto con la celda gris oscuro es $B\,\overline{C}\,\overline{D}E$.

Combinando estos términos obtenemos la suma de productos simplificada siguiente:

$$X = D\overline{E} + \overline{B}CE + \overline{A}B\overline{D} + B\overline{C}\,\overline{D}E$$

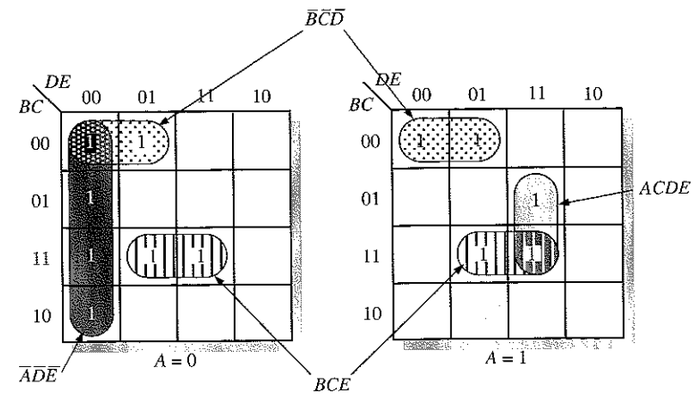
EJEMPLO 4.34 Utilizar un mapa de Karnaugh para minimizar la siguiente expresión estándar de la suma de productos de 5 variables:

$$X = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}\,\overline{E} + \overline{A}\,\overline{B}C\overline{D}\,\overline{E} + \overline{A}BC\overline{D}\,\overline{E} + \overline{A}B\overline{C}\,\overline{D}\,\overline{E} + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}E + \overline{A}BC\overline{D}E + \overline{A}BCDE + A\overline{B}\,\overline{C}\,\overline{D}\,\overline{E} + A\overline{B}\,\overline{C}\,\overline{D}E + ABC\overline{D}E + ABCDE + A\overline{B}CDE$$

Solución. En la Figura 4.43, se traslada la suma de productos al mapa de Karnaugh y se realizan las agrupaciones indicando los términos correspondientes. Combinando estos términos se obtiene la siguiente expresión suma de productos minimizada:

$$X = \overline{A}\,\overline{D}\,\overline{E} + \overline{B}\,\overline{C}\,\overline{D} + BCE + ACDE$$

FIGURA 4.43



Problema relacionado. Minimizar la siguiente expresión:

$$Y = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}\,\overline{E} + \overline{A}\,\overline{B}C\overline{D}\,\overline{E} + \overline{A}B\overline{C}\,\overline{D}\,\overline{E} + \overline{A}B\overline{C}\,\overline{D}E + A\overline{B}\,\overline{C}\,\overline{D}\,\overline{E} + A\overline{B}C\overline{D}\,\overline{E} + ABC\overline{D}\,\overline{E} + AB\overline{C}\,\overline{D}\,\overline{E} + \overline{A}\,\overline{B}\,\overline{C}D\overline{E} + \overline{A}\,\overline{B}CD\overline{E} + \overline{A}BCD\overline{E} + \overline{A}B\overline{C}D\overline{E} + A\overline{B}\,\overline{C}D\overline{E} + A\overline{B}CD\overline{E} + ABCD\overline{E} + AB\overline{C}D\overline{E}$$

REVISIÓN DE LA SECCIÓN 4.11

1. ¿Por qué un mapa de Karnaugh de 5 variables requiere 32 celdas?
2. ¿Cuál es la expresión representada por un mapa de Karnaugh de cinco variables en el que cada celda contiene un 1?

4.12 ■ APLICACIÓN A LOS SISTEMAS DIGITALES

Los displays de 7 segmentos se utilizan en casi todo, desde la instrumentación en automóviles hasta los medidores de impedancia. El sistema de control y recuento de pastillas, que ha sido el objetivo de nuestras aplicaciones de los capítulos anteriores, tiene dos displays de 7 segmentos. Estos displays se utilizan junto con circuitos lógicos que decodifican un número BCD y activan los dígitos adecuados del display. En esta sección, nos vamos a centrar en el diseño, con un número mínimo de puertas, de un decodificador lógico para ilustrar las aplicaciones de las expresiones booleanas y de los mapas de Karnaugh.

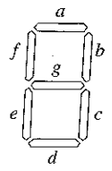
Al finalizar esta sección, el lector deberá ser capaz de:

■ Conocer el método de diseño de circuitos lógicos. ■ Explicar el funcionamiento de un display de 7 segmentos. ■ Traducir un determinado requisito en una expresión booleana y/o en su tabla de verdad y, posteriormente, en un circuito lógico que tenga un número de puertas mínimo utilizando un mapa de Karnaugh.

El display de 7 segmentos

La Figura 4.44 muestra un display común formado por siete elementos o segmentos. Excitando determinadas combinaciones de estos segmentos, se pueden obtener cada uno de los diez dígitos decimales. La Figura 4.45 muestra este tipo de display digital para cada uno de los diez dígitos, en el que se utiliza un segmento gris oscuro para indicar cuál está excitado. Para generar un 1, se excitan los segmentos *b* y *c*; para producir un 2, se excitan los segmentos *a, b, g, e* y *d*, y así sucesivamente.

FIGURA 4.44
Disposición de los segmentos en un display de 7 segmentos.



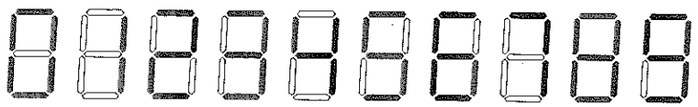
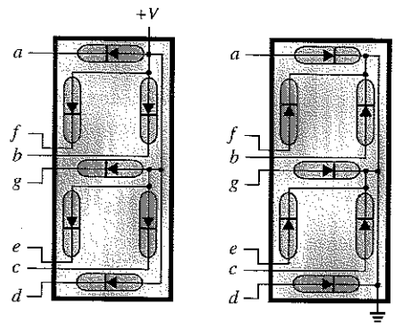


FIGURA 4.45
Display para dígitos decimales mediante un dispositivo de 7 segmentos.

Displays de LED. Un tipo muy común de display de 7 segmentos es el de diodos emisores de luz (light-emitting diode, **LED**), colocados como se muestra en la Figura 4.46. Cada segmento es un LED que emite luz cuando lo atraviesa una corriente eléctrica. En la Figura 4.46(a), la configuración en ánodo común requiere un circuito de excitación, que proporcione un nivel de tensión bajo para activar un determinado segmento. Cuando se aplica un nivel bajo a la entrada de un segmento, el LED se enciende y circula corriente a su través. En la Figura 4.46(b), la configuración en cátodo común requiere un circuito de excitación que proporcione un nivel de tensión alto para activar un cierto segmento. Cuando se aplica un nivel ALTO a la entrada del segmento, el LED se enciende y circula corriente a su través.

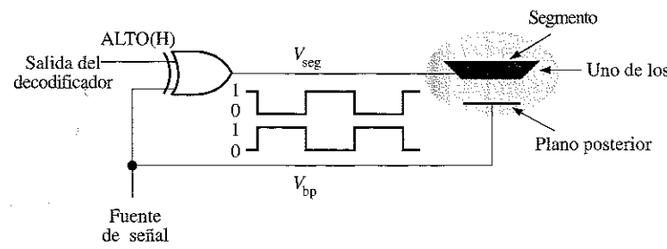


(a) Ánodo común (b) Cátodo común

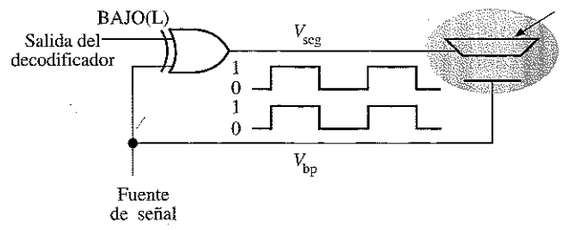
FIGURA 4.46
Configuraciones del display de 7 segmentos.

Displays de LCD. Otro tipo común de displays de 7 segmentos es el de cristal líquido, (**LCD**, liquid crystal display). Los LCD funcionan polarizando la luz de forma que un segmento que no está activado refleja la luz incidente, por lo que se ilumina. Un segmento activado no refleja la luz incidente y, por tanto, permanece oscuro. Los LCD consumen mucha menos potencia que los LED, pero no se pueden ver en la oscuridad, mientras que los LED sí.

Los LCD se excitan con una señal de tensión de baja frecuencia (de 30 Hz a 60 Hz) que se aplica entre el segmento y un elemento común denominado *plano posterior* (backplane). El funcionamiento básico se expone a continuación. La Figura 4.47 muestra una señal cuadrada que se usa como fuente de señal. Cada segmento del display es excitado por una puerta OR-exclusiva con una entrada conectada a la salida del decodificador/excitador de 7 segmentos, y la otra entrada conectada a la fuente de señal. Cuando la salida del decodificador/excitador está a nivel ALTO (1), la salida de la puerta OR-exclusiva actúa como un inversor para la señal de fuente, como se muestra en la Figura 4.47(a). Se puede verificar esto, revisando la tabla de verdad de una puerta OR-exclusiva. La tensión resultante entre el segmento LCD y el plano posterior es también una onda cuadrada, ya que cuando $V_{seg} = 1$, $V_{bp} = 0$, y viceversa. La diferencia de tensión enciende el segmento.



(a) Segmento activado (on)



(b) Segmento no activado (off)

FIGURA 4.47
Funcionamiento básico de un LCD.

Cuando la salida del decodificador/excitador es un nivel BAJO (0), la salida de la puerta OR-exclusiva deja pasar la señal de fuente no invertida, como se muestra en la Figura 4.47(b). La diferencia de tensión entre el segmento y el plano posterior es 0, ya que $V_{seg} = V_{bp}$. Esta condición hace que el segmento se apague.

Para excitar a los LCD, la tecnología TTL no es recomendable, dado que su tensión a nivel bajo es normalmente de unas décimas de voltio, la cual crea una componente continua a través del LCD que degrada su funcionamiento. Por tanto, en las aplicaciones con LCD se usa más la tecnología CMOS.

Lógica de decodificación de segmentos

Cada segmento se utiliza para varios dígitos decimales, pero ninguno de ellos se emplea para representar los diez dígitos; por tanto, cada segmento tiene que activarse mediante su propio circuito de decodificación que detecta la aparición de cualquier número en el que haya que usar ese segmento. A partir de las Figuras 4.44 y 4.45, se determinan los segmentos que hay que activar para representar cada uno de los dígitos, los cuales se enumeran en la Tabla 4.9.

TABLA 4.9
Segmentos activados para cada dígito decimal.

Dígito	Segmentos activados
0	*a, b, c, d, e, f*
1	*b, c*
2	*a, b, d, e, g*
3	*a, b, c, d, g*
4	*b, c, f, g*
5	*a, c, d, f, g*
6	*a, c, d, e, f, g*
7	*a, b, c*
8	*a, b, c, d, e, f, g*
9	*a, b, c, d, f, g*

Tabla de verdad de la lógica de segmentos. La lógica de decodificación de segmentos requiere cuatro entradas en código decimal binario (BCD) y siete salidas, una para cada segmento del display, como se indica en el diagrama de bloques de la Figura 4.48. La tabla de verdad de salida múltiple, que se muestra en la Tabla 4.10, corresponde en realidad a siete tablas de verdad, que podrían separarse en una tabla por segmento. Si aparece un 1 en las columnas de salida de la tabla, indica que el segmento está activado.

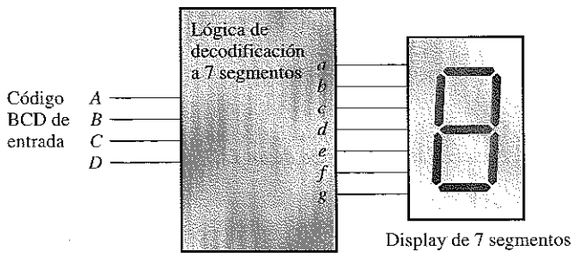


FIGURA 4.48
Diagrama de bloques de la lógica y el display de 7 segmentos.

TABLA 4.10
Tabla de verdad para la lógica de 7 segmentos.

Dígito decimal	Entradas D	C	B	A	Salidas de segmentos a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
10	1	0	1	0	X	X	X	X	X	X	X
11	1	0	1	1	X	X	X	X	X	X	X
12	1	1	0	0	X	X	X	X	X	X	X
13	1	1	0	1	X	X	X	X	X	X	X
14	1	1	1	0	X	X	X	X	X	X	X
15	1	1	1	1	X	X	X	X	X	X	X

Salida = 1 significa que el segmento está activado; Salida = 0 significa que el segmento está desactivado; Salida = X significa indiferente

Puesto que el código BCD no incluye los valores binarios 1010, 1011, 1100, 1101, 1110 y 1111, estas combinaciones no van nunca a aparecer en las entradas y pueden, por tanto, tratarse como condiciones indiferentes (X), como se indica en la tabla de verdad. Para coincidir con la mayoría de fabricantes de circuitos integrados, en esta aplicación una *A* representa el bit menos significativo y una *D* indica el bit más significativo.

Expresiones booleanas de la lógica de segmentos. A partir de la tabla de verdad, se puede escribir para cada segmento una expresión suma de productos o producto de sumas estándar. Por ejemplo, la suma de productos estándar para el segmento *a* es:

$$a = \overline{D}\,\overline{C}\,\overline{B}\,\overline{A} + \overline{D}\,\overline{C}B\overline{A} + \overline{D}\,\overline{C}BA + \overline{D}C\overline{B}\overline{A} + \overline{D}CB\overline{A} + \overline{D}CBA + D\overline{C}\,\overline{B}\,\overline{A} + D\overline{C}\,\overline{B}A$$

mientras que la suma de productos estándar para el segmento *e* es:

$$e = \overline{D}\,\overline{C}\,\overline{B}\,\overline{A} + \overline{D}\,\overline{C}B\overline{A} + \overline{D}CB\overline{A} + D\overline{C}\,\overline{B}\,\overline{A}$$

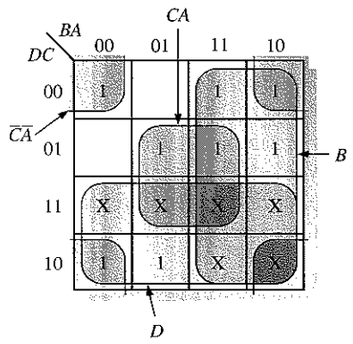
Se pueden desarrollar de la misma manera las expresiones para el resto de segmentos. Como puede verse, la expresión del segmento *a* contiene ocho productos, mientras que la expresión del segmento *e* contiene cuatro productos, que representan cada una de las entradas BCD que activan cada segmento. Esto significa que la implementación de la suma de productos estándar de la lógica del segmento *a* requiere un circuito AND-OR formado por ocho puertas AND de 4 entradas y una puerta OR de 8 entradas. La implementación de la lógica del segmento *e* requiere cuatro puertas AND de 4 entradas y una puerta OR de 4 entradas. En ambos casos, se necesitan cuatro inversores para generar el complemento de cada variable.

Minimización mediante el mapa de Karnaugh de la lógica de segmentos. Vamos a comenzar obteniendo una expresión suma de productos mínima para el segmento *a*. En la Figura 4.49 se muestra un mapa de Karnaugh correspondiente al segmento *a*. Los pasos que hay que seguir son los siguientes:

Paso 1. Los 1s de la Tabla 4.10 se pasan directamente al mapa de Karnaugh.
Paso 2. Se introducen en el mapa todas las condiciones "indiferentes" (X).
Paso 3. Se agrupan los 1s como se muestra. Se utilizan las condiciones "indiferentes" y superposiciones de celdas para conseguir los grupos más grandes posibles.
Paso 4. Se escribe el término producto mínimo para cada grupo y se suman para obtener la expresión suma de productos mínima.

No hay que olvidar que las condiciones "indiferentes" no tienen por qué incluirse en un grupo, pero en este caso se utilizan todas ellas. También hay que fijarse en que los 1s de las celdas de las esquinas se agrupan con condiciones indiferentes utilizando la adyacencia cíclica.

Expresión de la suma de productos estándar:
$\overline{D}C\overline{B}\overline{A} + \overline{D}C\overline{B}A + \overline{D}CB\overline{A} + \overline{D}CBA + D\overline{C}\overline{B}\overline{A} + D\overline{C}\overline{B}A + DC\overline{B}\overline{A} + DC\overline{B}A$



Suma de productos mínima: $D + B + CA + \overline{C}\overline{A}$

FIGURA 4.49
Minimización de la expresión lógica del segmento a mediante el mapa de Karnaugh.

Implementación mínima de la lógica del segmento a. La expresión mínima suma de productos a partir del mapa de Karnaugh de la Figura 4.49 para la lógica del segmento *a* es:

$$D + B + CA + \overline{C}\,\overline{A}$$

Esta expresión puede ser implementada mediante dos puertas AND de 2 entradas, una puerta OR de 4 entradas y dos inversores, como se muestra en la Figura 4.50. Compare este circuito con la implementación de la expresión estándar del segmento *a* vista anteriormente. Comprobará que el número de puertas e inversores se ha reducido de trece a cinco, disminuyendo significativamente el número de interconexiones necesarias.

La lógica mínima necesaria para cada uno de los restantes seis segmentos (*b, c, d, e, f* y *g*) puede obtenerse mediante un método similar.

■ PRÁCTICAS DE SISTEMAS

- ***Actividad 1.*** Determinar la lógica mínima para el segmento *b*.
- ***Actividad 2.*** Determinar la lógica mínima para el segmento *c*.
- ***Actividad 3.*** Determinar la lógica mínima para el segmento *d*.
- ***Actividad 4.*** Determinar la lógica mínima para el segmento *e*.
- ***Actividad 5.*** Determinar la lógica mínima para el segmento *f*.
- ***Actividad 6.*** Determinar la lógica mínima para el segmento *g*.
- ***Actividad 7.*** Determinar si existe algún producto duplicado en las expresiones de los siete segmentos, y, en caso afirmativo, eliminar todas las puertas AND duplicadas, menos una, de la implementación global.
- ***Actividad 8.*** Dibujar un diagrama lógico para el decodificador de 7 segmentos, combinando todos los circuitos lógicos de los segmentos individuales y eliminando las puertas e inversores duplicados.
- ***Actividad 9.*** Especificar mediante su referencia los circuitos integrados requeridos para implementar el diagrama lógico de la actividad 8, utilizando sólo aquellos que se han visto hasta ahora. Utilizar, cuando sea posible, todas las puertas disponibles del chip para que el número de circuitos integrados sea mínimo.

La lógica de decodificación de segmentos se encuentra en el archivo SA04.EWB del CD-ROM EWB. Utilizarlo para simular una prueba de este circuito lógico.

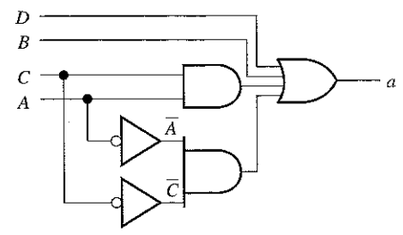


FIGURA 4.50
Implementación lógica mínima del segmento a de un display de 7 segmentos.

Simplificación de circuitos lógicos con salida múltiple. Una vez que se ha obtenido la expresión lógica mínima para cada segmento individual, todavía es posible realizar simplificaciones adicionales, eliminando cualquier término producto duplicado o de una única variable complementada. Por ejemplo, si la expresión suma de productos mínima del segmento *a* tiene un término que aparece también en, por ejemplo, el segmento *b*, entonces el producto duplicado puede implementarse mediante una única puerta AND, cuya salida esté conectada a las entradas de las puertas OR de cada segmento. De manera similar, el complemento de una variable sólo necesita ser implementado mediante un inversor, conectando posteriormente la salida del inversor donde la necesitemos, siempre que no se exceda su *fan-out*.

REVISIÓN DE LA SECCIÓN 4.12

1. ¿Cuál es la ventaja de usar condiciones "indiferentes" en una determinada aplicación?
2. ¿Cuántos términos duplicados se pueden encontrar en la implementación lógica de la actividad 7?
3. ¿Cuántas puertas e inversores se pueden eliminar como resultado de la duplicación en la actividad 8?

RESUMEN

- En la Figura 4.51 se muestran los símbolos y las expresiones booleanas de salida para un inversor y puertas de dos entradas.

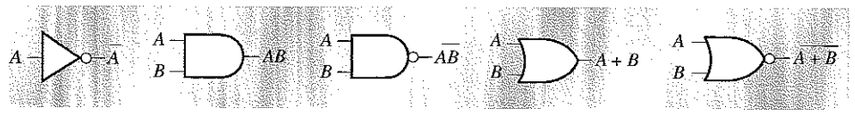


FIGURA 4.51

- Leyes conmutativas: $A + B = B + A$; $AB = BA$
- Leyes asociativas: $A + (B + C) = (A + B) + C$; $A(BC) = (AB)C$
- Ley distributiva: $A(B + C) = AB + AC$
- Reglas booleanas:
  1. $A + 0 = A$
  2. $A + 1 = 1$
  3. $A \cdot 0 = 0$
  4. $A \cdot 1 = A$
  5. $A + A = A$
  6. $A + \overline{A} = 1$
  7. $A \cdot A = A$
  8. $A \cdot \overline{A} = 0$
  9. $\overline{\overline{A}} = A$
  10. $A + AB = A$
  11. $A + \overline{A}B = A + B$
  12. $(A + B)(A + C) = A + BC$
- Teoremas de DeMorgan:
  1. El complemento de un producto es igual a la suma de los complementos de cada término del producto:
  $$\overline{XY} = \overline{X} + \overline{Y}$$
  2. El complemento de una suma es igual al producto de los complementos de cada término de la suma:
  $$\overline{X + Y} = \overline{X}\,\overline{Y}$$
- En la Figura 4.52 se muestran los mapas de Karnaugh para 3 y 4 variables. Un mapa de cinco variables se forma a partir de dos tablas de 4 variables.

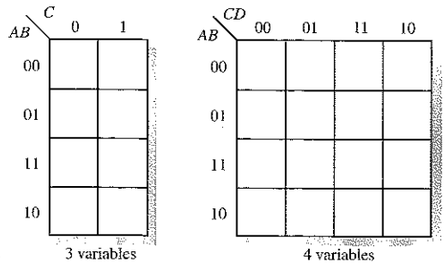


FIGURA 4.52

PALABRAS CLAVE

Estos términos también se encuentran en el glosario del final del libro.

Adyacencia Característica de las celdas en un mapa de Karnaugh en el que sólo se cambia una variable de una celda a otra inmediata a ella por cualquiera de sus cuatro lados.

Celda Una zona de un mapa de Karnaugh que representa una única combinación de variables.

Dominio Todas las variables de una expresión booleana.

Expresión booleana Ordenación de variables y operadores lógicos utilizada para expresar el funcionamiento de un circuito lógico.

Indiferente Combinación de literales de entrada que no pueden ocurrir y que se utilizan como 1s o 0s en un mapa de Karnaugh.

Ley asociativa En la suma (operación OR) y multiplicación (operación AND) de tres o más variables, el orden en que se agrupan las variables no altera el resultado.

Ley conmutativa En la suma (OR) o multiplicación (AND) de dos variables, el orden en el que las variables se suman o multiplican no altera el resultado.

Ley distributiva Ley que dice que si sumamos (operación OR) varias variables y luego multiplicamos (operación AND) el resultado por una única variable, lo que nos queda es equivalente a multiplicar (AND) la variable aislada por cada una de las otras variables y luego sumar (OR) todos los términos.

Literal Una variable o el complemento de una variable.

Mapa de Karnaugh Disposición de celdas que representa las combinaciones de literales en una expresión booleana y que se utiliza para la simplificación sistemática de la expresión.

Producto de sumas Expresión booleana que consiste simplemente en multiplicar (operación AND) términos suma (operación OR).

Suma de productos Expresión booleana que consiste simplemente en sumar (operación OR) términos producto (operación AND).

Término producto Producto booleano de dos o más literales equivalente a una operación AND.

Término suma Suma booleana de dos o más literales equivalente a la operación OR.

Variable Símbolo utilizado para representar una magnitud lógica que puede tener tanto un valor 1 como 0; generalmente se designa mediante una letra cursiva.

AUTOTEST

1. El complemento de una variable siempre es:
   (a) 0 **(b)** 1 **(c)** igual a la variable **(d)** el inverso de la variable
2. La expresión booleana $A + \overline{B} + C$ es:
   (a) un término suma **(b)** un literal
   (c) un término producto **(d)** un término complementado

3. La expresión booleana $A\overline{B}C\overline{D}$ es:

(a) un término suma **(b)** un término producto
(c) un literal **(d)** siempre 1

4. El dominio de la expresión $A\overline{B}C\overline{D} + A\overline{B} + \overline{C}D + B$ es:

(a) A y D **(b)** Sólo B **(c)** A, B, C y D **(d)** ninguno de los anteriores

5. De acuerdo con la ley conmutativa de la adición:

(a) $AB = BA$ **(b)** $A = A + A$ **(c)** $A + (B + C) = (A + B) + C$ **(d)** $A + B = B + A$

6. De acuerdo con la ley asociativa de la multiplicación:

(a) $B = BB$ **(b)** $A(BC) = (AB)C$ **(c)** $A + B = B + A$ **(d)** $B + B(B + 0)$

7. De acuerdo con la ley distributiva:

(a) $A(B + C) = AB + AC$ **(b)** $A(BC) = ABC$ **(c)** $A(A + 1) = A$ **(d)** $A + AB = A$

8. ¿Cuál de las siguientes no es una regla válida del álgebra booleana?

(a) $A + 1 = 1$ **(b)** $A = \overline{A}$ **(c)** $AA = A$ **(d)** $A + 0 = A$

9. ¿Cuál de las siguientes reglas establece que si una entrada de una puerta AND es siempre 1, la salida es igual a la otra entrada?

(a) $A + 1 = 1$ **(b)** $A + A = A$ **(c)** $A \cdot A = A$ **(d)** $A \cdot 1 = A$

10. De acuerdo con los teoremas de DeMorgan, ¿cuáles de las siguientes igualdades son correctas?:

(a) $\overline{AB} = \overline{A} + \overline{B}$ **(b)** $\overline{XYZ} = \overline{X} + \overline{Y} + \overline{Z}$ **(c)** $\overline{A+B+C} = \overline{A}\,\overline{B}\,\overline{C}$ **(d)** todas las anteriores

11. La expresión booleana $X = AB + CD$ representa:

(a) dos operaciones OR multiplicadas (AND)
(b) una puerta AND de cuatro entradas
(c) dos operaciones AND sumadas (OR)
(d) una operación OR-exclusiva

12. Un ejemplo de una suma de productos es:

(a) $A + B(C + D)$ **(b)** $\overline{A}B + A\overline{C} + A\overline{B}C$ **(c)** $(\overline{A} + B + C)(A + \overline{B} + C)$ **(d)** (a) y (b)

13. Un ejemplo de un producto de sumas es:

(a) $A(B + C) + A\overline{C}$ **(b)** $(A + B)(\overline{A} + B + \overline{C})$ **(c)** $\overline{A} + \overline{B} + BC$ **(d)** (a) y (b)

14. Un ejemplo de una expresión suma de productos estándar es:

(a) $\overline{A}B + A\overline{B}C + AB\overline{D}$ **(b)** $A\overline{B}C + A\overline{C}D$ **(c)** $A\overline{B} + \overline{A}B + AB$ **(d)** $A\overline{B}C\overline{D} + \overline{A}B + \overline{A}$

15. Un mapa de Karnaugh de tres variables tiene:

(a) 8 celdas **(b)** 3 celdas **(c)** 16 celdas **(d)** 4 celdas

16. En un mapa de Karnaugh de 4 variables, un término producto de 2 variables se produce por:

(a) un grupo de 2 celdas de 1s
(b) un grupo de 4 celdas de 1s
(c) un grupo de 8 celdas de 1s
(d) un grupo de 4 celdas de 0s

17. En un mapa de Karnaugh, el agrupar los 0s da lugar a:

(a) un producto de sumas
(b) una suma de productos
(c) una condición indiferente
(d) un circuito lógico AND-OR

18. Un mapa de Karnaugh de 5 variables tiene:

(a) 16 celdas **(b)** 32 celdas **(c)** 64 celdas

■ PROBLEMAS

SECCIÓN 4.1 Operaciones y expresiones booleanas

1. Utilizando notación booleana, escribir una expresión que sea 1 siempre que una o más de sus variables (A, B, C y D) sean 1.

2. Escribir una expresión que sea 1 sólo si todas sus variables (A, B, C, D y E) son 1.

3. Escribir una expresión que sea 1 cuando una o más de sus variables (A, B y C) son 0.

4. Evaluar las siguientes operaciones:

(a) $0 + 0 + 1$ **(b)** $1 + 1 + 1$ **(c)** $1 \cdot 0 \cdot 0$
(d) $1 \cdot 1 \cdot 1$ **(e)** $1 \cdot 0 \cdot 1$ **(f)** $1 \cdot 1 + 0 \cdot 1 \cdot 1$

5. Encontrar los valores de las variables que hacen que cada término producto sea 1 y que cada suma sea 0.

(a) AB **(b)** $A\overline{B}C$ **(c)** $A + B$ **(d)** $\overline{A} + B + \overline{C}$
(e) $\overline{A} + \overline{B} + C$ **(f)** $\overline{A} + B$ **(g)** $A\overline{B}\,\overline{C}$

6. Encontrar el valor de X para todos los posibles valores de las variables.

(a) $X = (A + B)C + B$ **(b)** $X = \overline{(A+B)}C$ **(c)** $X = A\overline{B}C + AB$
(d) $X = (A + B)(\overline{A} + B)$ **(e)** $X = (A + BC)(\overline{B} + \overline{C})$

SECCIÓN 4.2 Leyes y reglas del álgebra de Boole

7. Identificar la ley del álgebra booleana en que está basada cada una de las siguientes igualdades:

(a) $A\overline{B} + CD + A\overline{C}D + B = B + A\overline{B} + A\overline{C}D + CD$

(b) $AB\overline{C}D + \overline{A}B\overline{C} = D\overline{C}BA + \overline{C}B\overline{A}$

(c) $AB(CD + E\overline{F} + GH) = ABCD + ABE\overline{F} + ABGH$

8. Identificar las reglas booleanas en que se basan las siguientes igualdades:

(a) $\overline{A\overline{B} + CD} + \overline{E}\overline{F} = AB + CD + \overline{E}\overline{F}$ **(b)** $A\overline{A}B + AB\overline{C} + AB\overline{B} = AB\overline{C}$
(c) $A(BC + BC) + AC = A(BC) + AC$ **(d)** $AB(C + \overline{C}) + AC = AB + AC$
(e) $A\overline{B} + A\overline{B}C = A\overline{B}$ **(f)** $ABC + \overline{A}\overline{B} + \overline{A}\overline{B}CD = ABC + \overline{A}\overline{B} + D$

SECCIÓN 4.3 Teoremas de DeMorgan

9. Aplicar los teoremas de DeMorgan a cada una de las expresiones siguientes:

(a) $\overline{A + \overline{B}}$ **(b)** $\overline{\overline{A}B}$ **(c)** $\overline{A + B + C}$ **(d)** $\overline{AB\overline{C}}$
(e) $\overline{A(B + \overline{C})}$ **(f)** $\overline{A\overline{B} + \overline{C}D}$ **(g)** $\overline{A\overline{B} + CD}$ **(h)** $\overline{(A + \overline{B})(\overline{C} + D)}$

10. Aplicar los teoremas de DeMorgan a cada expresión:

(a) $\overline{A\overline{B}(C + \overline{D})}$ **(b)** $\overline{AB(CD + \overline{EF})}$

(c) $\overline{(A + \overline{B} + C + \overline{D})} + \overline{ABC\overline{D}}$ **(d)** $\overline{(\overline{A} + B + C + D)(AB\overline{C}D)}$

(e) $\overline{A\overline{B}(CD + \overline{E}F)(\overline{A}B + \overline{C}D)}$

11. Aplicar los teoremas de DeMorgan a las siguientes expresiones:

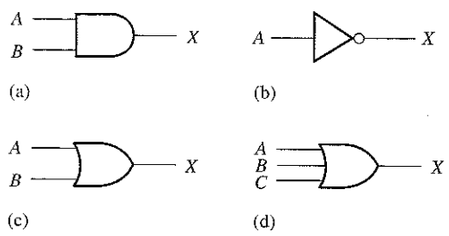
(a) $\overline{(AB\overline{C})(EFG)} + \overline{(HIJ)(KLM)}$ **(b)** $\overline{(A + B\overline{C} + CD)} + \overline{\overline{B}C}$

(c) $\overline{(A + B)(\overline{C} + D)(E + F)(\overline{G} + H)}$

SECCIÓN 4.4 Análisis booleano de los circuitos lógicos

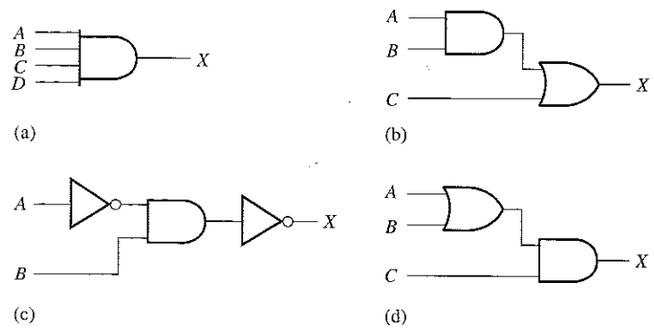
12. Escribir la expresión booleana para cada puerta lógica de la Figura 4.53.

FIGURA 4.53



13. Escribir la expresión booleana para cada uno de los circuitos lógicos de la Figura 4.54.

FIGURA 4.54



14. Dibujar el circuito lógico representado por cada una de las siguientes expresiones:

(a) $A + B + C$ **(b)** ABC **(c)** $AB + C$ **(d)** $AB + CD$

15. Dibujar el circuito lógico representado por las siguientes expresiones booleanas:

(a) $A\overline{B} + \overline{A}B$ **(b)** $AB + \overline{A}\,\overline{B} + \overline{A}BC$

(c) $\overline{A}B(C + \overline{D})$ **(d)** $A + B[C + D(B + \overline{C})]$

16. Construir la tabla de verdad para cada una de las siguientes expresiones booleanas:

(a) $A + B$ **(b)** AB **(c)** $AB + BC$

(d) $(A + B)C$ **(e)** $(A + B)(\overline{B} + C)$

SECCIÓN 4.5 Simplificación mediante el álgebra de Boole

17. Mediante las técnicas del álgebra booleana, simplificar las siguientes expresiones lo máximo posible:

(a) $A(A + B)$ **(b)** $A(\overline{A} + AB)$ **(c)** $BC + \overline{B}C$

(d) $A(A + \overline{A}B)$ **(e)** $A\overline{B}C + \overline{A}BC + \overline{A}\,\overline{B}C$

18. Mediante el álgebra booleana, simplificar las siguientes expresiones:

(a) $(A + \overline{B})(A + C)$ **(b)** $\overline{A}B + \overline{A}B\overline{C} + \overline{A}BCD + \overline{A}B\overline{C}\,\overline{D}E$

(c) $AB + \overline{A}B\overline{C} + A$ **(d)** $(A + \overline{A})(AB + AB\overline{C})$

(e) $AB + (\overline{A} + \overline{B})C + AB$

19. Mediante el álgebra booleana, simplificar las siguientes expresiones:

(a) $BD + B(D + E) + \overline{D}(D + F)$ **(b)** $\overline{A}\,\overline{B}C + \overline{(A + B + \overline{C})} + \overline{A}\,\overline{B}\,\overline{C}\,D$

(c) $(B + BC)(B + \overline{B}C)(B + D)$ **(d)** $ABCD + AB(\overline{CD}) + (\overline{AB})CD$

(e) $ABC[AB + \overline{C}(BC + AC)]$

20. Determinar cuáles de los circuitos lógicos de la Figura 4.55 son equivalentes.

SECCIÓN 4.6 Formas estándar de las expresiones booleanas

21. Convertir las siguientes expresiones en sumas de productos:

(a) $(A + B)(C + \overline{B})$ **(b)** $(A + \overline{B}C)C$ **(c)** $(A + C)(AB + AC)$

22. Convertir las siguientes expresiones en sumas de productos:

(a) $AB + CD(A\overline{B} + CD)$ **(b)** $AB(\overline{B}\,\overline{C} + BD)$ **(c)** $A + B[AC + (B + \overline{C})D]$

23. Definir el dominio de cada suma de productos del problema 21 y convertir la expresión a su forma estándar.

24. Convertir cada suma de productos del problema 22 a su forma estándar.

25. Determinar el valor binario de cada término en las sumas de productos estándar del problema 23.

26. Determinar el valor binario de cada término en las sumas de productos estándar del problema 24.

27. Convertir cada expresión suma de productos estándar del problema 23 en producto de sumas estándar.

28. Convertir cada expresión suma de productos estándar del problema 24 en producto de sumas estándar.

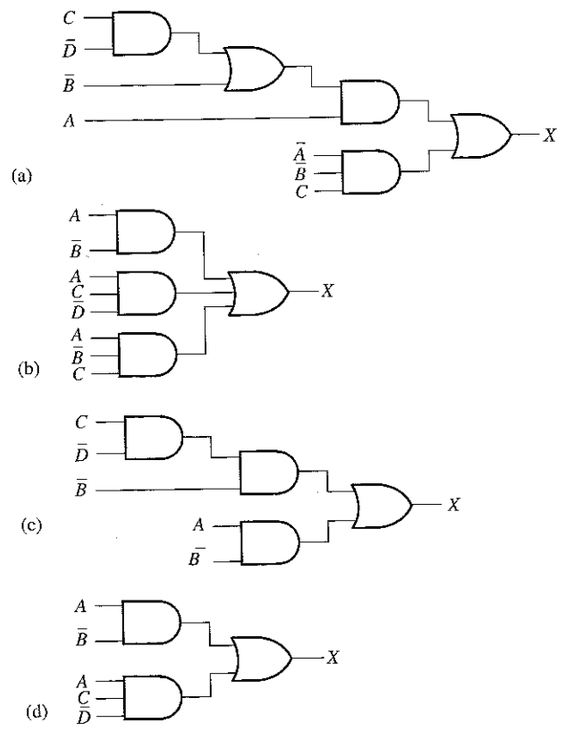


FIGURA 4.55

SECCIÓN 4.7 Expresiones booleanas y tablas de verdad

29. Desarrollar la tabla de verdad de cada una de las siguientes expresiones suma de productos estándar:

(a) $A\overline{B}C + \overline{A}B\overline{C} + ABC$ **(b)** $\overline{X}YZ + \overline{X}\,\overline{Y}Z + XY\overline{Z} + X\overline{Y}Z + \overline{X}YZ$

30. Desarrollar la tabla de verdad de cada una de las siguientes expresiones suma de productos estándar:

(a) $\overline{A}B\overline{C}D + \overline{A}BC\overline{D} + A\overline{B}\,\overline{C}D + \overline{A}B\overline{C}\,\overline{D}$ **(b)** $WXYZ + WXY\overline{Z} + \overline{W}XYZ + W\overline{X}YZ + WX\overline{Y}Z$

31. Desarrollar la tabla de verdad de cada una de las siguientes sumas de productos:

(a) $\overline{A}B + AB\overline{C} + \overline{A}\,\overline{C} + A\overline{B}C$ **(b)** $\overline{X} + Y\overline{Z} + WZ + X\overline{Y}Z$

32. Desarrollar la tabla de verdad de cada uno de los siguientes productos de sumas estándar:

(a) $(\overline{A} + \overline{B} + \overline{C})(A + B + C)(A + \overline{B} + C)$

(b) $(\overline{A} + B + \overline{C} + D)(A + \overline{B} + C + \overline{D})(A + \overline{B} + \overline{C} + D)\,(\overline{A} + B + C + \overline{D})$

33. Desarrollar la tabla de verdad de cada producto de sumas estándar siguiente:

(a) $(A + B)(A + C)(A + B + C)$ **(b)** $(A + \overline{B})(A + \overline{B} + \overline{C})(B + C + \overline{D})(\overline{A} + B + \overline{C} + D)$

34. Para cada tabla de verdad de la Figura 4.56, obtener una expresión suma de productos estándar y un producto de sumas estándar.

FIGURA 4.56

ABC	X
000	0
001	1
010	0
011	0
100	1
101	1
110	0
111	1

(a)

ABC	X
000	0
001	0
010	0
011	0
100	0
101	1
110	1
111	1

(b)

$ABCD$	X
0000	1
0001	1
0010	0
0011	1
0100	0
0101	1
0110	1
0111	0
1000	0
1001	1
1010	0
1011	0
1100	1
1101	0
1110	0
1111	0

(c)

$ABCD$	X
0000	0
0001	0
0010	1
0011	0
0100	1
0101	1
0110	0
0111	1
1000	0
1001	0
1010	0
1011	1
1100	1
1101	0
1110	0
1111	1

(d)

SECCIÓN 4.8 Mapas de Karnaugh

35. Dibujar un mapa de Karnaugh de 3 variables y etiquetar cada celda según su valor binario.

36. Dibujar un mapa de Karnaugh de 4 variables y etiquetar cada celda según su valor binario.

37. Escribir los términos producto estándar correspondientes a cada celda de un mapa de Karnaugh de 3 variables.

SECCIÓN 4.9 Minimización de una suma de productos mediante el mapa de Karnaugh

38. Utilizando un mapa de Karnaugh encontrar la suma de productos mínima para cada una de las expresiones:

(a) $\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + A\overline{B}C$ **(b)** $AC(\overline{B} + C)$ **(c)** $\overline{A}(BC + B\overline{C}) + A(BC + B\overline{C})$

(d) $\overline{A}\,\overline{B}\,\overline{C} + A\overline{B}\,\overline{C} + \overline{A}B\overline{C} + AB\overline{C}$

39. Utilizar un mapa de Karnaugh para simplificar cada una de las expresiones a su forma suma de productos mínima:

(a) $\overline{A}\,\overline{B}\,\overline{C} + A\overline{B}C + \overline{A}BC + AB\overline{C}$ **(b)** $AC[\overline{B} + B(B + \overline{C})]$ **(c)** $DE\overline{F} + \overline{D}E\overline{F} + \overline{D}\,\overline{E}\,\overline{F}$

40. Expandir cada expresión a su forma suma de productos estándar:

(a) $AB + A\overline{B}C + ABC$ **(b)** $A + BC$ **(c)** $A\overline{B}\,\overline{C}D + AC\overline{D} + B\overline{C}D + \overline{A}BC\overline{D}$

(d) $A\overline{B} + A\overline{B}\,\overline{C}D + CD + B\overline{C}D + ABCD$

41. Minimizar cada expresión del problema 40 mediante un mapa de Karnaugh.

42. Utilizar el mapa de Karnaugh para reducir cada expresión a su forma suma de productos mínima:

(a) $A + B\overline{C} + CD$

(b) $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + ABCD + ABC\overline{D}$

(c) $\overline{A}B(\overline{C}\,\overline{D} + \overline{C}D) + AB(\overline{C}\,\overline{D} + \overline{C}D) + A\overline{B}\,\overline{C}D$

(d) $(\overline{A}\,\overline{B} + A\overline{B})(CD + C\overline{D})$

(e) $\overline{A}\,\overline{B} + A\overline{B} + \overline{C}\,\overline{D} + C\overline{D}$

43. Reducir la función especificada en la tabla de verdad de la Figura 4.57 a su forma suma de productos mínima mediante el mapa de Karnaugh.

44. Utilizar el mapa de Karnaugh para implementar la forma suma de productos mínima de la función lógica especificada en la tabla de verdad de la Figura 4.58.

45. Resolver el problema 44 para una situación en que las seis últimas combinaciones binarias no están permitidas.

Entradas			Salida
A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

FIGURA 4.57

Entradas				Salida
A	B	C	D	X
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

FIGURA 4.58

SECCIÓN 4.10 Minimización de un producto de sumas mediante el mapa de Karnaugh

46. Utilizar el mapa de Karnaugh para encontrar la expresión producto de sumas mínima de las expresiones:

(a) $(A + B + C)(\overline{A} + \overline{B} + \overline{C})(A + \overline{B} + C)$ **(b)** $(X + \overline{Y})(\overline{X} + Z)(X + \overline{Y} + \overline{Z})(\overline{X} + \overline{Y} + Z)$

(c) $A(B + \overline{C})(\overline{A} + C)(A + \overline{B} + C)(\overline{A} + B + \overline{C})$

47. Utilizar el mapa de Karnaugh para simplificar cada expresión a su forma producto de sumas mínima:

(a) $(A + \overline{B} + C + \overline{D})(\overline{A} + B + \overline{C} + D)(\overline{A} + \overline{B} + \overline{C} + \overline{D})$

(b) $(X + \overline{Y})(W + \overline{Z})(\overline{X} + \overline{Y} + \overline{Z})(W + X + Y + Z)$

48. Para la función especificada en la tabla de verdad de la Figura 4.57, determinar el producto de sumas mínimo mediante el mapa de Karnaugh.

49. Determinar el producto de sumas mínimo para la función de la tabla de verdad de la Figura 4.58.

50. Convertir cada uno de los siguientes productos de sumas a su mínima expresión suma de productos, utilizando el mapa de Karnaugh.

(a) $(A + \overline{B})(A + \overline{C})(\overline{A} + \overline{B} + C)$

(b) $(\overline{A} + B)(\overline{A} + \overline{B} + \overline{C})(B + \overline{C} + D)(A + \overline{B} + C + \overline{D})$

SECCIÓN 4.11 Mapa de Karnaugh de cinco variables

51. Minimizar las siguientes sumas de productos utilizando el mapa de Karnaugh.

$$X = \overline{A}B\overline{C}D\overline{E} + \overline{A}\,B\,\overline{C}DE + A\overline{B}\,\overline{C}DE + AB\overline{C}\,\overline{D}\,\overline{E} + \overline{A}B\overline{C}D\overline{E} + \overline{A}BC\overline{D}E + \overline{A}\,B\,\overline{C}\,\overline{D}\,\overline{E} + \overline{A}\,B\overline{C}DE + AB\overline{C}D\overline{E} + AB\overline{C}DE$$

52. Aplicar el mapa de Karnaugh para minimizar la siguiente suma de productos:

$$A = \overline{V}WXYZ + V\overline{W}XYZ + VW\overline{X}YZ + VWX\overline{Y}Z + VWXY\overline{Z} + \overline{V}\,\overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z} + \overline{V}\,\overline{W}\,\overline{X}YZ + \overline{V}\,\overline{W}X\overline{Y}\,\overline{Z} + \overline{V}\,\overline{W}X\overline{X}\,\overline{Y}\,\overline{Z}$$

SECCIÓN 4.12 Aplicación a los sistemas digitales

53. Si es necesario elegir un tipo de display para trabajar bajo condiciones de baja luminosidad, ¿cuál se seleccionaría: un display de 7 segmentos de diodos LED o de cristal líquido? ¿por qué?

54. Explicar por qué los códigos 1010, 1011, 1100, 1101, 1110 y 1111 pertenecen a la categoría de condiciones "indiferentes" en las aplicaciones con displays de 7 segmentos.

55. Para el segmento *b*, ¿cuántas puertas e inversores menos se necesitan para implementar la suma de productos mínima con respecto a la suma de productos estándar?

56. Repetir el problema 55 para la lógica de los segmentos *c* hasta *g*.

Problemas especiales de diseño

57. La lógica del segmento *a* de la Figura 4.50 produce una salida a nivel ALTO para activar el segmento, ocurriendo lo mismo para el resto de los segmentos. Si se utiliza un display de 7 segmentos que requiere un nivel BAJO para activar cada segmento, modificar adecuadamente la lógica del segmento.

58. Rediseñar la lógica del segmento *a* utilizando un producto de sumas mínimo. ¿Cuál es más sencilla, la suma de productos mínima o el producto de sumas mínimo?

59. Repetir el problema 58 para los segmentos *b* hasta *g*.

60. Resumir los resultados del rediseño que se ha hecho en los problemas 58 y 59, y recomendar el mejor diseño en función del mínimo número de circuitos integrados. Especificar los tipos de CI que se utilizarían.

Localización de averías con el banco de trabajo electrónico

61. Abrir el archivo PR04_61.EWB y observar el funcionamiento del circuito lógico utilizando la configuración de prueba existente. Determinar si existe o no un fallo.

62. Abrir el archivo PR04_62.EWB y observar el funcionamiento del circuito lógico utilizando la configuración de prueba existente. Determinar si existe o no un fallo.

63. Abrir el archivo PR04_63.EWB y observar el funcionamiento del circuito lógico utilizando la configuración de prueba existente. Determinar si existe o no un fallo.

■ RESPUESTAS A LAS REVISIONES DE CADA SECCIÓN

SECCIÓN 4.1

1. $\overline{A} = \overline{0} = 1$

2. $A = 1, B = 1, C = 0; \overline{A} + \overline{B} + C = \overline{1} + \overline{1} + 0 = 0 + 0 + 0 = 0$

3. $A = 1, B = 0, C = 1; A\overline{B}C = 1 \cdot \overline{0} \cdot 1 = 1 \cdot 1 \cdot 1 = 1$

SECCIÓN 4.2

1. $A + (B + C + D) = (A + B + C) + D$

2. $A(B + C + D) = AB + AC + AD$

SECCIÓN 4.3

1. **(a)** $\overline{A\overline{B}C} + \overline{(\overline{D} + E)} = \overline{A} + \overline{B} + \overline{C} + D\overline{E}$

(b) $\overline{(A + B)C} = \overline{A}\,\overline{B} + \overline{C}$

(c) $\overline{A + B + C} + \overline{\overline{D}E} = \overline{A}\,\overline{B}\,\overline{C} + D + \overline{E}$

SECCIÓN 4.4

1. $(C + D)B + A$

2. Tabla de verdad abreviada: la expresión es 1 cuando A es 1 o B y C son 1, o cuando B y D son 1. La expresión es 0 para el resto de combinaciones de las variables.

SECCIÓN 4.5

1. **(a)** $A + AB + A\overline{B}C = A$

(b) $(\overline{A} + B)C + ABC = C(\overline{A} + B)$

(c) $A\overline{B}C(BD + CDE) + A\overline{C} = A(\overline{C} + \overline{B}DE)$

2. **(a)** *Original:* 2 puertas AND, 1 puerta OR, 1 inversor. *Simplificada*: sin puertas (conexión directa).

(b) *Original:* 2 puertas OR, 2 puertas AND, 1 inversor. *Simplificada*: 1 puerta OR, 1 puerta AND, 1 inversor.

(c) *Original:* 5 puertas AND, 2 puertas OR, 2 inversores. *Simplificada*: 2 puertas AND, 1 puerta OR, 2 inversores.

SECCIÓN 4.6

1. **(a)** suma de productos

(b) producto de sumas estándar

(c) suma de productos estándar

(d) producto de sumas

2. **(a)** $AB\overline{C}\,\overline{D} + AB\overline{C}D + ABC\overline{D} + ABCD + \overline{A}B\overline{C}D + \overline{A}BCD + \overline{A}\,\overline{B}C\overline{D} + \overline{A}BC\overline{D}$
   (c) estándar
3. **(b)** estándar
   (d) $(A + \overline{B} + \overline{C})(A + \overline{B} + C)(A + B + \overline{C})\ (A + B + C)$

SECCIÓN 4.7

1. $2^5 = 32$
2. $0110 \longrightarrow \overline{W}XY\overline{Z}$
3. $1100 \longrightarrow \overline{W} + \overline{X} + Y + Z$

SECCIÓN 4.8

1. **(a)** celda superior izquierda: 000
   (b) celda inferior derecha: 101
   (c) celda inferior izquierda: 100
   (d) celda superior derecha: 001
2. **(a)** celda superior izquierda: $\overline{X}\,\overline{Y}\,\overline{Z}$
   (b) celda inferior derecha: $X\overline{Y}Z$
   (c) celda inferior izquierda: $X\overline{Y}\,\overline{Z}$
   (d) celda superior derecha: $\overline{X}\,\overline{Y}Z$
3. **(a)** celda superior izquierda: 0000
   (b) celda inferior derecha: 1010
   (c) celda inferior izquierda: 1000
   (d) celda superior derecha: 0010
4. **(a)** celda superior izquierda: $\overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z}$
   (b) celda inferior derecha: $W\overline{X}Y\overline{Z}$
   (c) celda inferior izquierda: $W\overline{X}\,\overline{Y}\,\overline{Z}$
   (d) celda superior derecha: $\overline{W}\,\overline{X}Y\overline{Z}$

SECCIÓN 4.9

1. Mapa de 8 celdas para 3 variables. Mapa de 16 celdas para 4 variables.
2. $AB + B\overline{C} + \overline{A}\,\overline{B}C$
3. **(a)** $\overline{A}\,\overline{B}\,\overline{C} + \overline{A}BC + ABC + AB\overline{C}$
   (b) $\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + ABC + A\overline{B}\,\overline{C} + A\overline{B}C$
   (c) $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}B\overline{C}\,\overline{D} + \overline{A}B\overline{C}D + \overline{A}BCD + \overline{A}BC\overline{D} + A\overline{B}\,\overline{C}D + A\overline{B}CD$
   (d) $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}B\overline{C}\,\overline{D} + AB\overline{C}\,\overline{D} + A\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\overline{B}\overline{C}D + AB\overline{C}D + A\overline{B}CD + \overline{A}\,\overline{B}C\overline{D} + \overline{A}BC\overline{D} + ABC\overline{D} + A\overline{B}C\overline{D}$

SECCIÓN 4.10

1. Cuando se pasa a un mapa de Karnaugh un producto de sumas, los 0s se introducen en las celdas cuyos valores hacen que el término suma estándar sea igual a 0; cuando se pasa a un mapa de Karnaugh una suma de productos, los 1s se colocan en las celdas que tienen los mismos valores que los términos producto.
2. 0 en la celda 1011: $\overline{A} + B + \overline{C} + \overline{D}$
3. 1 en la celda 0010: $\overline{A}\,\overline{B}C\overline{D}$

SECCIÓN 4.11

1. Existen 32 combinaciones de las 5 variables ($2^5 = 32$).
2. $X = 1$, ya que la función es 1 para todas las posibles combinaciones de las 5 variables.

SECCIÓN 4.12

1. Los términos "indiferentes" simplifican las expresiones lógicas.
2. Debería haberse encontrado siete términos duplicados en el circuito lógico de segmentos.
3. Se han eliminado ocho puertas AND y once inversores.

■ RESPUESTAS A LOS PROBLEMAS RELACIONADOS DE LOS EJEMPLOS

4.1 $\overline{A} + B = 0$ cuando $A = 1$ y $B = 0$. **4.2** $\overline{A}\,\overline{B} = 1$ cuando $A = 0$ y $B = 0$. **4.3** XYZ

4.4 $W + X + Y + Z$ **4.5** $ABC\overline{D}\,\overline{E}$ **4.6** $(A + \overline{B} + \overline{C}D)\overline{E}$

4.7 $\overline{AB\overline{C}D} = \overline{A} + \overline{B} + \overline{C} + \overline{D}$ **4.8** $A\overline{B}$ **4.9** CD **4.10** $AB\overline{C} + \overline{A}C + \overline{A}\,\overline{B}$

4.11 $\overline{A} + \overline{B} + \overline{C}$ **4.12** $\overline{A}B\overline{C} + AB + A\overline{C} + A\overline{B} + \overline{B}\,\overline{C}$

4.13 $W\overline{X}YZ + W\overline{X}Y\overline{Z} + W\overline{X}\,\overline{Y}\,\overline{Z} + \overline{W}\,\overline{X}Y\overline{Z} + WX\overline{Y}Z + WX\overline{Y}\,\overline{Z}$

4.14 011, 101, 110, 010, 111. Sí. **4.15** $(A + \overline{B} + C)(A + \overline{B} + \overline{C})(A + B + C)(\overline{A} + B + C)$

4.16 010, 100, 001, 111, 011. Sí.

4.17 Las expresiones suma de productos y producto de sumas son equivalentes.

4.18 Véase la Tabla 4.11

4.19 Véase la Tabla 4.12.

TABLA 4.11

A	*B*	*C*	*X*
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

TABLA 4.12

A	*B*	*C*	*X*
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

4.20 Las expresiones suma de productos y producto de sumas son equivalentes.

4.21 Véase la Figura 4.59.

4.22 Véase la Figura 4.60.

4.23 Véase la Figura 4.61.

4.24 Véase la Figura 4.62.

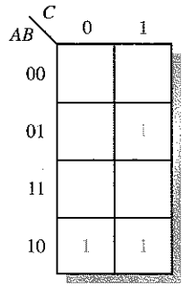


FIGURA 4.59

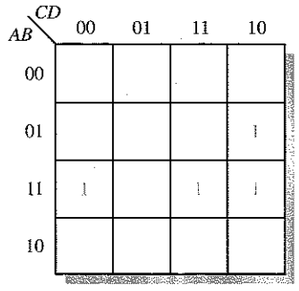


FIGURA 4.60

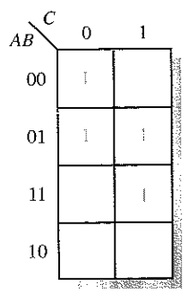


FIGURA 4.61

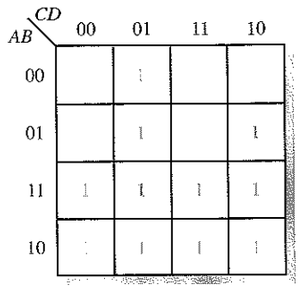


FIGURA 4.62

4.25 Ninguna otra forma.

4.26 $X = B + \overline{A}C + A\overline{C}D + C\overline{D}$

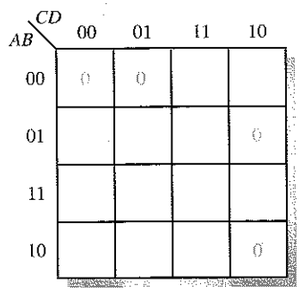
4.27 $X = \overline{D} + A\overline{B}C + B\overline{C} + \overline{A}B$

4.28 $Q = X + Y$

4.29 $Q = \overline{X}\,\overline{Y}\,\overline{Z} + W\overline{X}Z + \overline{W}YZ$

4.30 Véase la Figura 4.63.

FIGURA 4.63



4.31 $Q = (X + \overline{Y})(X + \overline{Z})(\overline{X} + Y + Z)$

4.32 $Q = (\overline{X} + \overline{Y} + Z)(\overline{W} + \overline{X} + Z)(W + X + Y + Z)(W + \overline{X} + Y + \overline{Z})$

4.33 $Q = \overline{Y}\,\overline{Z} + \overline{X}\,\overline{Z} + \overline{W}Y + \overline{X}\,\overline{Y}Z$

4.34 $Y = \overline{D}\,\overline{E} + \overline{A}\,\overline{E} + \overline{B}\,\overline{C}\,\overline{E}$

■ RESPUESTAS AL AUTOTEST

1. (d) **2.** (a) **3.** (b) **4.** (c) **5.** (d) **6.** (b) **7.** (a) **8.** (b)
9. (d) **10.** (d) **11.** (c) **12.** (b) **13.** (b) **14.** (c) **15.** (a) **16.** (c)
17. (a) **18.** (b)

5

LÓGICA COMBINACIONAL

5.1 Circuitos lógicos combinacionales especiales
5.2 Implementación de la lógica combinacional
5.3 La propiedad universal de las puertas NAND y NOR
5.4 Lógica combinacional utilizando las puertas NAND y NOR
5.5 Funcionamiento con trenes de impulsos
5.6 Localización de averías
5.7 Aplicación a los sistemas digitales

WWW Simulación por computadora de circuitos digitales:
El tutorial de Electronics Workbench (EWB) se encuentra en la dirección http://www.prenhall.com/floyd.

■ OBJETIVOS DEL CAPÍTULO

- ❑ Analizar los circuitos lógicos combinacionales básicos, tales como AND-OR, AND-OR-Inversor, OR-exclusiva, NOR-exclusiva y otras redes combinacionales generales.
- ❑ Utilizar los circuitos AND-OR y AND-OR-Inversor para implementar expresiones como suma de productos y producto de sumas.
- ❑ Escribir la expresión booleana de salida de cualquier circuito lógico combinacional.
- ❑ Desarrollar la tabla de verdad a partir de la expresión de salida de un circuito lógico combinacional.
- ❑ Utilizar el mapa de Karnaugh para expandir una expresión de salida, que contenga variables suprimidas en una suma de productos completa.
- ❑ Diseñar un circuito lógico combinacional para una expresión booleana de salida dada.
- ❑ Diseñar un circuito lógico combinacional para una tabla de verdad determinada.
- ❑ Simplificar un circuito lógico combinacional a su forma mínima.
- ❑ Utilizar puertas NAND para implementar cualquier función lógica combinacional.
- ❑ Utilizar puertas NOR para implementar cualquier función lógica combinacional.
- ❑ Localizar fallos en los circuitos lógicos.
- ❑ Localizar fallos en los circuitos lógicos utilizando el trazado de señales y el análisis de las formas de onda.

■ INTRODUCCIÓN

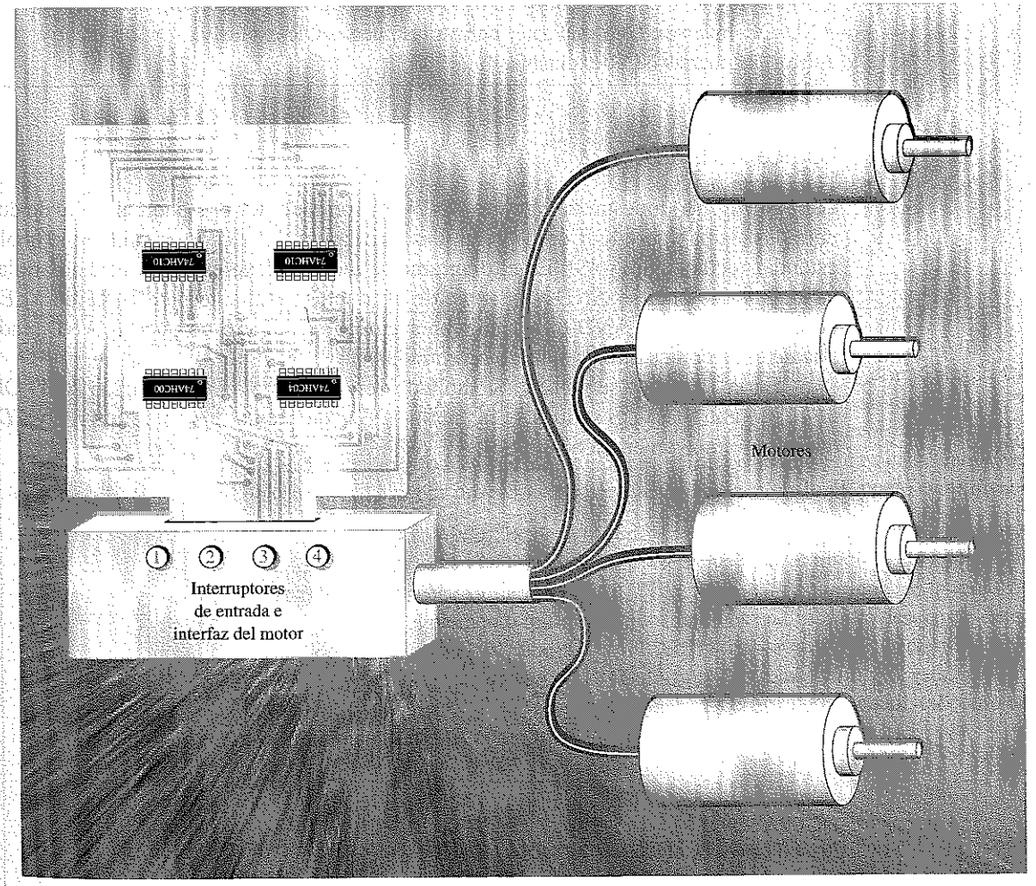
En los Capítulos 3 y 4, nos hemos ocupado de las puertas lógicas como elementos individuales y en sencillas combinaciones. Se han introducido las implementaciones de suma de productos y producto de sumas, que son las formas básicas de la lógica combinacional. Cuando se conectan puertas lógicas entre sí, con el fin de generar una determinada salida específica para determinadas combinaciones específicas de las variables de entrada, sin que haya implicado almacenamiento, el circuito resultante se califica como **lógica combinacional**. En la lógica combinacional, el nivel de salida depende siempre de la combinación de los niveles de entrada. Este capítulo amplía el material introducido en los capítulos anteriores y cubre el análisis, diseño y localización de fallos de diversos circuitos lógicos combinacionales.

DISPOSITIVOS ESPECÍFICOS

74HC58 74LS51 74LS54

■ APLICACIÓN A LOS SISTEMAS DIGITALES

Esta aplicación a un sistema digital ilustra los conceptos que se enseñan en este capítulo, mostrando cómo se puede usar la lógica combinacional para un propósito específico en una aplicación práctica. En la Sección 5.7, se usa un circuito digital para controlar una operación de tratamiento de la madera en una fábrica de muebles. El circuito digital funcional a partir de cuatro interruptores de entrada que sirven para controlar los cuatro motores que hacen funcionar al sistema. En función de la configuración de los interruptores, el circuito lógico determina qué motores deberían activarse y cuáles no, y desactiva el sistema si los interruptores no están correctamente configurados.



5.1 ■ CIRCUITOS LÓGICOS COMBINACIONALES ESPECIALES

En el Capítulo 4, hemos aprendido que las expresiones suma de productos se implementan con una puerta AND para cada término producto y una puerta OR para sumar todos los productos. Esta implementación de la suma de productos se llama lógica AND-OR y es la forma básica para realizar las funciones estándar booleanas. En esta sección, se estudian las combinaciones AND-OR y AND-OR-Inversor y las puertas OR-exclusiva y NOR-exclusiva, que realmente son una forma de la lógica AND-OR.

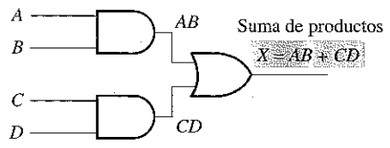
Al finalizar esta sección, el lector deberá ser capaz de:

■ Analizar y aplicar los circuitos AND-OR. ■ Analizar y aplicar los circuitos AND-OR-Inversor. ■ Analizar y aplicar las puertas OR-exclusiva. ■ Analizar y aplicar las puertas NOR-exclusiva.

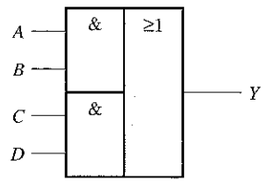
Lógica AND-OR

La Figura 5.1(a) muestra un circuito AND-OR formado por dos puertas AND de dos entradas y una puerta OR de dos entradas. La Figura 5.1(b) corresponde al símbolo estándar rectangular ANSI. En el diagrama se indica la expresión booleana para las salidas de las puertas AND y la suma de productos resultante para la salida X. En general, un circuito AND-OR puede tener cualquier número de puertas AND, cada una de ellas con cualquier número de entradas.

En la Tabla 5.1 se presenta la tabla de verdad para el circuito lógico AND-OR de cuatro entradas. Las salidas de la puerta AND intermedia (las columnas AB y CD) también se indican en dicha tabla.



(a) Diagrama lógico (símbolos distintivos del estándar ANSI)



(b) Símbolo rectangular del estándar ANSI

FIGURA 5.1
Ejemplo de circuito lógico AND-OR.

Un circuito AND-OR implementa directamente una suma de productos, suponiendo que se dispone de los complementos de las variables. La operación lógica del circuito AND-OR de la Figura 5.1 se enuncia como sigue:

En un circuito lógico AND-OR de 4 entradas, la salida X es un nivel ALTO (1) sólo si las dos entradas A y B están a nivel ALTO (1) o si las dos entradas C y D están a nivel ALTO (1).

TABLA 5.1
Tabla de verdad para el circuito lógico AND-OR de la Figura 5.1

TABLA 5.1

Entradas						Salida
A	*B*	*C*	*D*	*AB*	*CD*	*X*
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	1	1
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	1	1	0	1	1
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	0	1	1
1	1	0	0	1	0	1
1	1	0	1	1	0	1
1	1	1	0	1	0	1
1	1	1	1	1	1	1

EJEMPLO 5.1 En una determinada planta de procesamiento químico se emplea un elemento químico líquido en un proceso de fabricación. Dicho elemento químico se almacena en tres tanques diferentes. Un sensor de nivel en cada tanque genera una tensión a nivel ALTO cuando el nivel de líquido en el tanque cae por debajo de un punto especificado.

Diseñar un circuito para supervisar el nivel del elemento químico en cada tanque, que indique cuándo el nivel de dos tanques cualesquiera cae por debajo del punto especificado.

Solución. El circuito AND-OR de la Figura 5.2 tiene entradas procedentes de los sensores de los tanques A, B y C. La puerta AND G_1 comprueba el nivel en los tanques A y B, la puerta G_2 comprueba los tanques A y C, y la puerta G_3 comprueba los tanques B y C. Cuando el nivel del elemento químico en dos tanques cualesquiera desciende demasiado, una de las puertas AND tendrá a nivel ALTO ambas entradas, haciendo que su salida sea un nivel ALTO, por lo que la salida final X de la puerta OR estará a nivel ALTO. Esta salida a nivel ALTO se usa entonces para activar un indicador, tal como una alarma luminosa o audible, como muestra la figura.

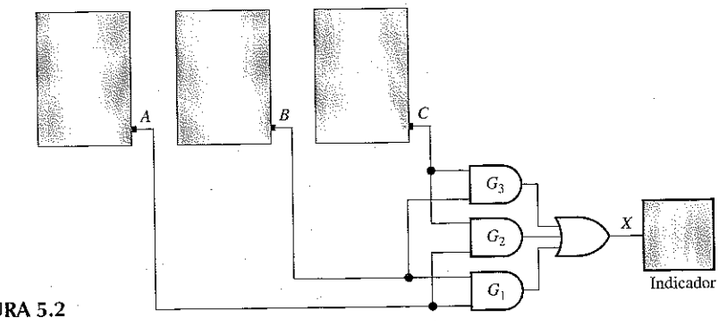
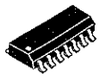
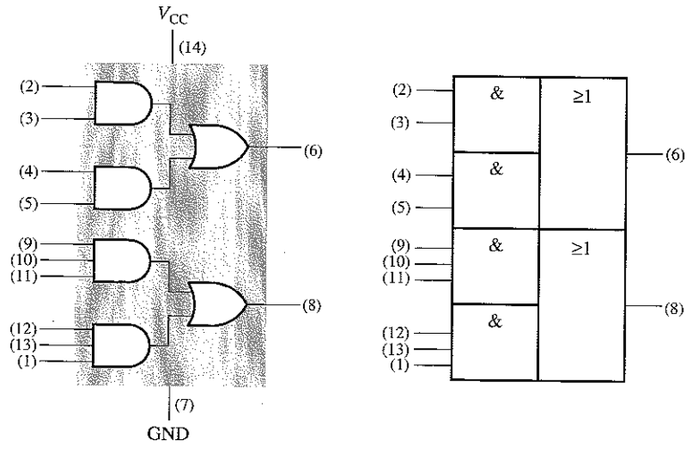


FIGURA 5.2

Problema relacionado. Escribir la expresión booleana en forma de suma de productos para el circuito lógico AND-OR de la Figura 5.2.



Un circuito integrado AND-OR específico. El 74HC58 es un ejemplo de un CI lógico AND-OR. Se trata de un dispositivo CMOS que contiene dos circuitos AND-OR separados en un único encapsulado. Uno de los circuitos tiene puertas AND de 2 entradas, y el otro tiene puertas AND de 3 entradas, como muestra la Figura 5.3(a). También se indica la numeración de los pines para los encapsulados de perfil bajo (SOIC). En la parte (b) se muestra el símbolo rectangular estándar ANSI.



(a) Diagrama lógico (b) Símbolo rectangular

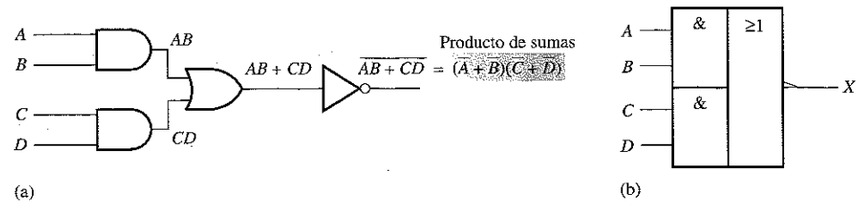
FIGURA 5.3
El 74HC58, doble AND-OR.

Circuito lógico AND-OR-Inversor

Cuando se complementa (invierte) la salida de un circuito AND-OR, se obtiene el circuito AND-OR-Inversor. Recuérdese que el circuito AND-OR implementa directamente la suma de productos. El producto de sumas puede implementarse con un circuito lógico AND-OR-Inversor. Esto se ilustra de la forma siguiente, partiendo del producto de sumas y desarrollando la expresión AND-OR-Inversor correspondiente.

$$X = (\overline{A} + \overline{B})(\overline{C} + \overline{D}) = (\overline{AB})(\overline{CD}) = \overline{\overline{(\overline{AB})(\overline{CD})}} = \overline{\overline{\overline{AB}} + \overline{\overline{CD}}} = \overline{AB + CD}$$

El diagrama lógico de la Figura 5.4(a) muestra un circuito AND-OR-Inversor y el desarrollo de la expresión de salida como producto de sumas. En la parte (b) de la figura se presenta el símbolo rectangular estándar ANSI. En general, un circuito AND-OR-Inversor puede tener cualquier número de puertas AND, y cada una de ellas puede tener cualquier número de entradas.



(a) (b)

FIGURA 5.4
Un circuito AND-OR-Inversor genera una salida que es un producto de sumas.

La operación lógica del circuito AND-OR-Inversor de la Figura 5.4 se define como:

En un circuito lógico AND-OR-Inversor, la entrada *X* es un nivel BAJO (0) si las dos entradas *A* y *B* están a nivel ALTO (1), o si las dos entradas *C* y *D* están a nivel ALTO (1).

Se puede desarrollar la tabla de verdad a partir de la Tabla 5.1, tabla de verdad del circuito AND-OR, cambiando simplemente, en la columna de salida, todos los 1s por 0s y todos los 0s por 1s.

EJEMPLO 5.2 Los sensores colocados en los tanques químicos del Ejemplo 5.1 se reemplazan por un nuevo modelo que genera una tensión a nivel BAJO en lugar de una tensión a nivel ALTO cuando el nivel de líquido en el tanque cae por debajo del punto crítico.

Modificar el circuito de la Figura 5.2 para trabajar con los diferentes niveles de entrada y generar una salida a nivel ALTO que active el indicador cuando el nivel de dos tanques caiga por debajo del punto crítico. Realizar el diagrama lógico.

Solución. Los sensores de los tanques *A*, *B* y *C* se conectan a las entradas del circuito AND-OR-Inversor, como se muestra en la Figura 5.5. La puerta AND G_1 comprueba el nivel en los tanques *A* y *B*, la puerta G_2 comprueba los tanques *A* y *C*, y la puerta G_3 comprueba los tanques *B* y *C*. Cuando el nivel del elemento químico en dos tanques cualesquiera desciende, al menos una de las entradas de cada una de las puertas AND estará a nivel BAJO, haciendo que su salida sea un nivel BAJO, por lo que la salida final *X* del inversor estará a nivel ALTO. Esta salida a nivel ALTO se usa entonces para activar un indicador.

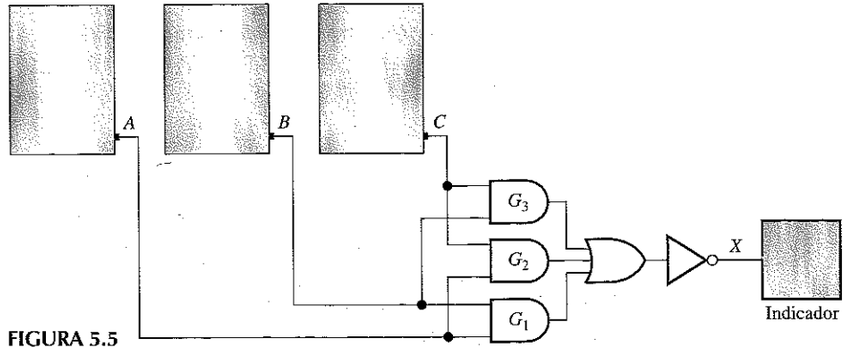
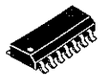


FIGURA 5.5

Problema relacionado. Escribir la expresión booleana para el circuito lógico AND-OR-Inversor de la Figura 5.5 y mostrar que la salida es un nivel ALTO (1) cuando dos entradas cualesquiera de entre A, B y C estén a nivel BAJO (0).



Circuitos integrados AND-OR-Inversor específicos. El 74LS51 y el 74LS54 son ejemplos de un circuito lógico AND-OR-Inversor. Ambos son dispositivos TTL Schottky de baja potencia, aunque también están disponibles en otras familias lógicas. El 74LS51 contiene dos circuitos AND-OR-Inversor separados, en un único encapsulado. Uno de los circuitos tiene puertas AND de 2 entradas y el otro circuito tiene puertas AND de 3 entradas, como muestra la Figura 5.6(a). El 74LS54, como muestra la Figura 5.6(b), contiene un único circuito AND-OR-Inversor, que tiene dos puertas AND de 2 entradas y dos puertas AND de 3 entradas. Obsérvese que la inversión se indica mediante un círculo en la salida de las puertas OR, lo que demuestra que la parte OR-Inversor del circuito es efectivamente una puerta NOR.

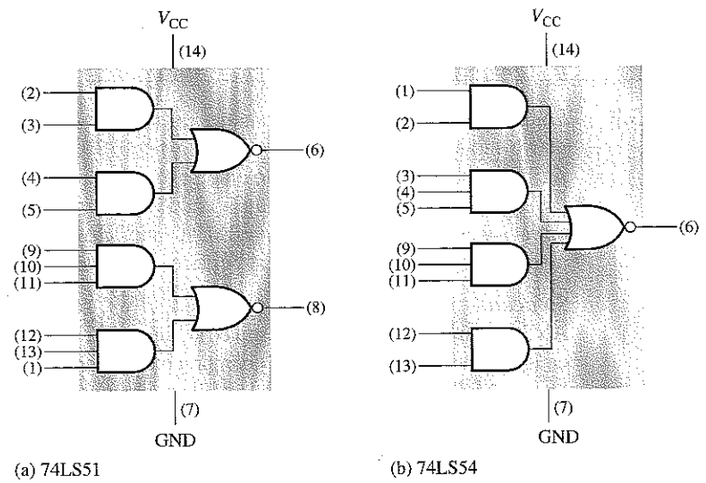


FIGURA 5.6
El 74LS51, doble AND-OR-Inversor de doble anchura y el 74LS54, AND-OR-Inversor de cuádruple anchura. El término doble anchura significa que hay dos puertas AND por dispositivo y cuádruple anchura que hay cuatro puertas AND por dispositivo.

Circuito lógico OR-exclusiva

En el Capítulo 3 se han introducido la puerta OR-exclusiva y el circuito integrado 74XX86. Aunque, debido a su importancia este circuito se considera como una puerta lógica con su propio símbolo distintivo, realmente es una combinación de dos puertas AND, una puerta OR y dos inversores, tal y como muestra la Figura 5.7(a). En las Figuras 5.7 (b) y (c) se presentan los dos símbolos lógicos estándar ANSI.

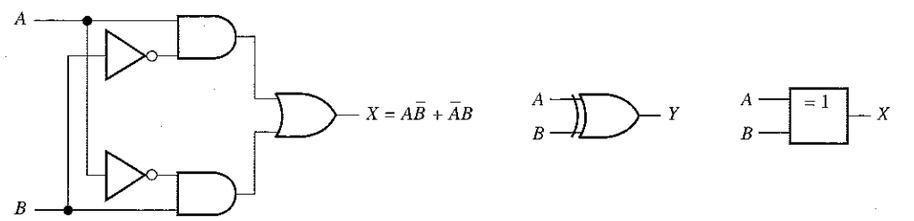


FIGURA 5.7
Diagrama lógico y símbolos del circuito OR-exclusiva

La expresión de salida para el circuito de la Figura 5.7 es

$$X = A\overline{B} + \overline{A}B$$

A partir de la evaluación de esta expresión se obtiene la tabla de verdad presentada en la Tabla 5.2. Obsérvese que la salida es un nivel alto sólo cuando las dos entradas están a niveles opuestos. A menudo se usa el operador especial OR-Exclusiva, $\oplus$, luego la expresión $X = A\overline{B} + \overline{A}B$ indica que "X es igual a A OR-Exclusiva B", y puede escribirse como:

$$X = A \oplus B$$

TABLA 5.2
Tabla de verdad para el circuito OR- Exclusiva.

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

Circuito lógico NOR-exclusiva

Como sabemos, el complemento de la función OR-exclusiva es NOR-exclusiva, la cual se obtiene del siguiente modo:

$$X = \overline{A\overline{B} + \overline{A}B} = \overline{(A\overline{B})(\overline{A}B)} = (\overline{A} + B)(A + \overline{B}) = \overline{A}\,\overline{B} + AB$$

Obsérvese que la salida X es un nivel ALTO sólo cuando las dos entradas, A y B, están al mismo nivel.

La función NOR-exclusiva puede implementarse invirtiendo la salida de un circuito OR-exclusiva, como muestra la Figura 5.8(a), o bien se puede implementar directamente a partir de la expresión $\overline{A}\,\overline{B} + AB$, como muestra la parte (b) de la figura.

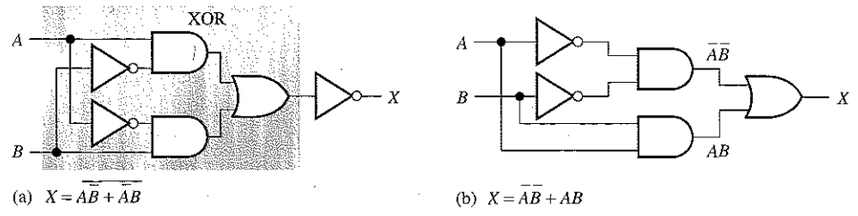


FIGURA 5.8
Dos formas equivalentes de implementar el circuito NOR-exclusiva

REVISIÓN DE LA SECCIÓN 5.1

1. Determinar la salida (1 o 0) de un circuito AND-OR-Inversor de 4 variables para cada una de las siguientes condiciones de entrada:
   (a) $A = 1, B = 0, C = 1, D = 0$
   (b) $A = 1, B = 1, C = 0, D = 1$
   (c) $A = 0, B = 1, C = 1, D = 1$
2. Determinar la salida (1 o 0) de una puerta OR-Exclusiva para cada una de las siguientes condiciones de entrada:
   (a) $A = 1, B = 0$ **(b)** $A = 1, B = 1$
   (c) $A = 0, B = 1$ **(d)** $A = 0, B = 0$
3. Desarrollar la tabla de verdad para un circuito lógico de 3 entradas con la expresión de salida
   $X = A\bar{B}C + \bar{A}BC + A\bar{B}\bar{C} + AB\bar{C} + ABC$.
4. Dibujar el diagrama lógico para un circuito NOR Exclusiva.

5.2 ■ IMPLEMENTACIÓN DE LA LÓGICA COMBINACIONAL

En esta sección se usan ejemplos para ilustrar cómo implementar un circuito lógico a partir de una expresión booleana o una tabla de verdad. También se tratará la minimización de un circuito lógico utilizando los métodos vistos en el Capítulo 4.

Al finalizar esta sección, el lector deberá ser capaz de:

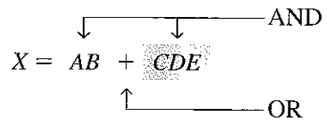
■ Implementar un circuito lógico a partir de una expresión booleana. ■ Implementar un circuito lógico a partir de una tabla de verdad. ■ Minimizar un circuito lógico.

Obtención del circuito lógico a partir de una expresión booleana

Examinemos la siguiente expresión booleana:

$$X = AB + CDE$$

Una rápida inspección revela que esta expresión consta de dos términos, *AB* y *CDE*, y que hay un total de cinco variables. El primer término es la multiplicación (AND) de *A y B*, y el segundo término es la multiplicación de *C, D y E*. Luego, los dos términos se suman (OR) para generar la salida *X*. Estas operaciones se indican en la siguiente estructura de la expresión:

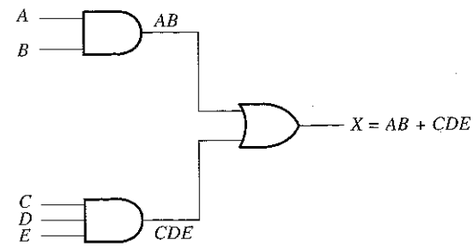


Obsérvese que, en esta expresión, las operaciones AND son dos términos individuales, *AB* y *CDE*, que deben efectuarse *antes* de aplicar la operación OR.

Para implementar esta expresión booleana se requiere una puerta AND de 2 entradas para obtener el término *AB*, y una puerta AND de tres entradas para generar el término *CDE*. Para combinar los dos términos obtenidos en las puertas AND se requiere una puerta OR de 2 entradas. El resultado se muestra en la Figura 5.9.

NOTAS INFORMÁTICAS Muchos programas de control requieren operaciones lógicas para ser ejecutados por una computadora. Un controlador es un programa que se usa con los periféricos de la computadora. Por ejemplo, un controlador de ratón requiere pruebas lógicas para determinar si se ha presionado un botón, y más operaciones lógicas para determinar si se ha movido horizontal o verticalmente. En el núcleo de un microprocesador se encuentra la unidad aritmético lógica (ALU), que realiza estas operaciones lógicas mediante instrucciones de programa. Toda la lógica descrita en este capítulo también puede ser realizada por la ALU, proporcionando las instrucciones adecuadas.

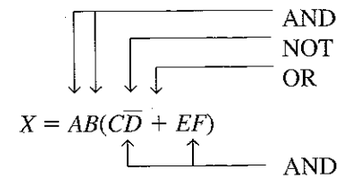
FIGURA 5.9
Circuito lógico para X = AB + CDE



Como ejemplo, vamos a implementar otra expresión:

$$X = AB(C\bar{D} + EF)$$

Si se divide esta expresión, vemos que los términos *A*, *B* y $C\bar{D} + EF$ se multiplican (AND). El término $C\bar{D} + EF$ está formado por la multiplicación (AND) de *C* y $\bar{D}$ y la multiplicación (AND) de *E* y *F*, y después estos dos términos se suman (OR). Esta estructura se expresa del siguiente modo:



Antes de generar la expresión completa, debe obtener el término $C\bar{D} + EF$. Para ello, previamente hay que disponer de los términos $C\bar{D}$ y *EF*, pero antes de obtener el término $C\bar{D}$ es necesario tener $\bar{D}$. Luego, como puede verse, las operaciones lógicas deben efectuarse en el orden adecuado.

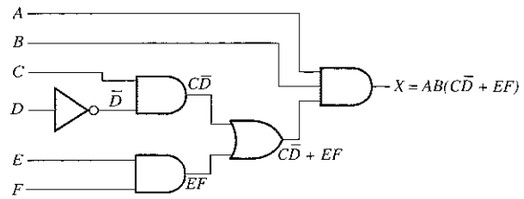
Las puertas lógicas que se requieren para implementar la expresión $X = AB(C\bar{D} + EF)$ son las siguientes:

1. Un inversor para obtener $\bar{D}$.
2. Dos puertas AND de 2 entradas para obtener $C\bar{D}$ y *EF*.
3. Una puerta OR de 2 entradas para obtener $C\bar{D} + EF$
4. Una puerta AND de 3 entradas para obtener *X*.

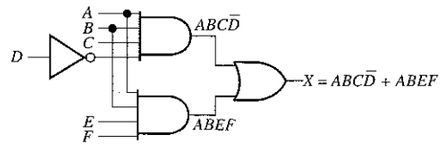
En la Figura 5.10(a) se muestra el circuito lógico correspondiente a esta expresión. Obsérvese que hay un máximo de tres puertas y un inversor entre una entrada y la salida del circuito (de la entrada *D* a la salida). A menudo, el retardo de propagación a través del circuito lógico es una consideración de gran importancia. Los retardos de propagación se suman, luego cuantos más inversores y puertas haya entre la entrada y la salida, mayor es el retardo de propagación.

A menos que un término intermedio, tal como $C\bar{D} + EF$ de la Figura 5.10(a), se requiera como salida para algún otro propósito, usualmente lo mejor es reducir el circuito a su suma de productos. La expresión se convierte en suma de productos como sigue, y el circuito resultante se muestra en la Figura 5.10(b).

$$AB(C\bar{D} + EF) = ABC\bar{D} + ABEF$$



(a)



(b) Implementación de la suma de productos del circuito del apartado (a)

FIGURA 5.10
Circuitos lógicos para la expresión $X = AB(C\overline{D} + EF) = ABC\overline{D} + ABEF$.

Obtención del circuito lógico a partir de la tabla de verdad

Si en lugar de partir de una expresión se parte de una tabla de verdad, puede escribirse la suma de productos que se obtiene de la tabla de verdad, y luego implementar el circuito lógico. La Tabla 5.3 especifica una función lógica.

TABLA 5.3

Entradas			Salida	
A	*B*	*C*	*X*	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\longrightarrow \overline{A}BC$
1	0	0	1	$\longrightarrow A\overline{B}\,\overline{C}$
1	0	1	0	
1	1	0	0	
1	1	1	0	

La expresión booleana suma de productos que se obtiene a partir de la tabla de verdad, haciendo la suma (OR) de los productos para los que X es 1 (filas sombreadas) es

$$X = \overline{A}BC + A\overline{B}\,\overline{C}$$

El primer término de la expresión se obtiene haciendo el producto lógico (AND) de las tres variables $\overline{A}$, $\overline{B}$ y C. El segundo término se forma aplicando la operación AND a las tres variables A, $\overline{B}$ y $\overline{C}$.

Las puertas lógicas requeridas para implementar esta expresión son las siguientes: tres inversores para obtener las variables $\overline{A}$, $\overline{B}$ y $\overline{C}$; dos puertas AND de 3 entradas para obtener los términos $\overline{A}BC$ y $A\overline{B}\overline{C}$, y una puerta OR de 2 entradas para conseguir la función de salida final, $\overline{A}BC + A\overline{B}\overline{C}$.

La Figura 5.11 ilustra la implementación de esta función lógica.

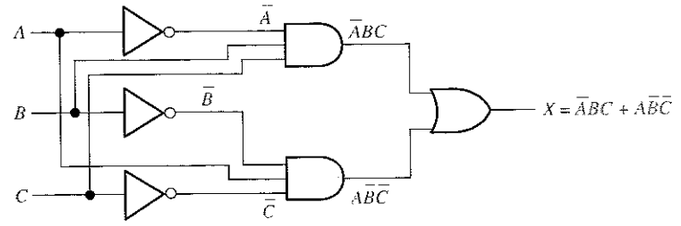


FIGURA 5.11
Circuito lógico para $X = \overline{A}BC + A\overline{B}\,\overline{C}$

EJEMPLO 5.3 Diseñar un circuito lógico para implementar la operación especificada en la tabla de verdad indicada en la Tabla 5.4.

TABLA 5.4

Entradas			Salida
A	*B*	*C*	*X*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Solución. Obsérvese que X = 1 sólo para tres de las condiciones de entrada. Por tanto, la expresión lógica es:

$$X = \overline{A}BC + A\overline{B}C + AB\overline{C}$$

Las puertas lógicas requeridas son tres inversores, tres puertas AND de 3 entradas y una puerta OR de 3 entradas. El circuito lógico es el que se muestra en la Figura 5.12.

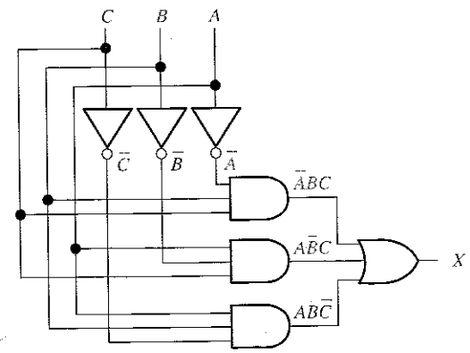


FIGURA 5.12

Problema relacionado. Determinar si el circuito lógico de la Figura 5.12 puede simplificarse.

EJEMPLO 5.4 Desarrollar un circuito lógico con cuatro variables de entrada que sólo genera un 1 en la salida cuando tres variables de entrada son 1.

Solución. Para cuatro variables, existen dieciséis posibles combinaciones; de éstas las que contienen tres 1s son las que se enumeran en la Tabla 5.5, junto con el correspondiente término producto.

TABLA 5.5

A	*B*	*C*	*D*	
0	1	1	1	$\longrightarrow \overline{A}BCD$
1	0	1	1	$\longrightarrow A\overline{B}CD$
1	1	0	1	$\longrightarrow AB\overline{C}D$
1	1	1	0	$\longrightarrow ABC\overline{D}$

Se aplica la operación OR a los productos para obtener la siguiente expresión:

$$X = \overline{A}BCD + A\overline{B}CD + AB\overline{C}D + ABC\overline{D}$$

Esta expresión se implementa con el circuito lógico AND-OR de la Figura 5.13.

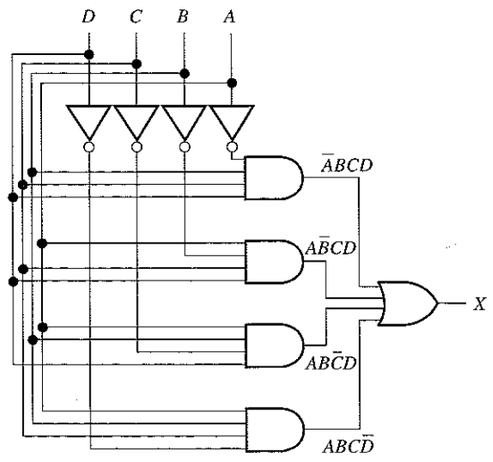


FIGURA 5.13

Problema relacionado. Determinar si el circuito lógico de la Figura 5.13 puede simplificarse.

EJEMPLO 5.5 Reducir el circuito lógico combinacional de la Figura 5.14 a una forma mínima.

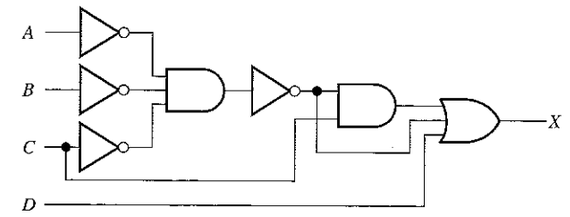


FIGURA 5.14

Solución. La expresión para la salida del circuito es:

$$X = (\overline{\overline{A}\,\overline{B}\,\overline{C}})C + \overline{\overline{A}\,\overline{B}\,\overline{C}} + D$$

Aplicando el teorema de DeMorgan y el álgebra booleana

$$\begin{aligned} X &= (\overline{\overline{A} + \overline{B} + \overline{C}})C + \overline{\overline{A}} + \overline{\overline{B}} + \overline{\overline{C}} + D \\ &= AC + BC + CC + A + B + C + D \\ &= AC + BC + C + A + B + \not{C} + D \\ &= C(A + B + 1) + A + B + D \\ X &= A + B + C + D \end{aligned}$$

El circuito simplificado es una puerta OR de cuatro entradas, como muestra la Figura 5.15.

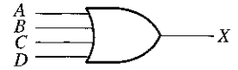


FIGURA 5.15

Problema relacionado. Verificar la expresión minimizada $(A + B + C + D)$ utilizando un mapa de Karnaugh.

EJEMPLO 5.6 Minimizar el circuito lógico combinacional de la Figura 5.16. Los inversores para las variables complementadas no se muestran.

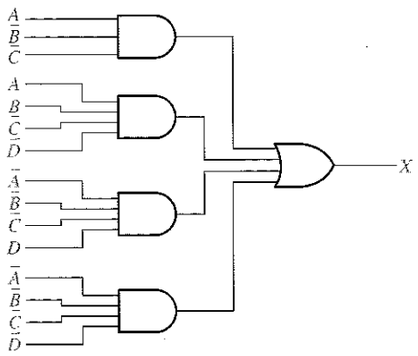
Solución. La expresión de salida es:

$$X = A\overline{B}\,\overline{C} + AB\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$$

Se añaden al primer término las variables omitidas D y $\overline{D}$.

$$\begin{aligned} X &= A\overline{B}\,\overline{C}(D + \overline{D}) + AB\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} \\ &= A\overline{B}\,\overline{C}D + A\overline{B}\,\overline{C}\,\overline{D} + AB\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} \end{aligned}$$

FIGURA 5.16



Esta suma de productos completa se traslada y simplifica en el mapa de Karnaugh de la Figura 5.17(a). La implementación simplificada se muestra en la parte (b). Los inversores no se muestran.

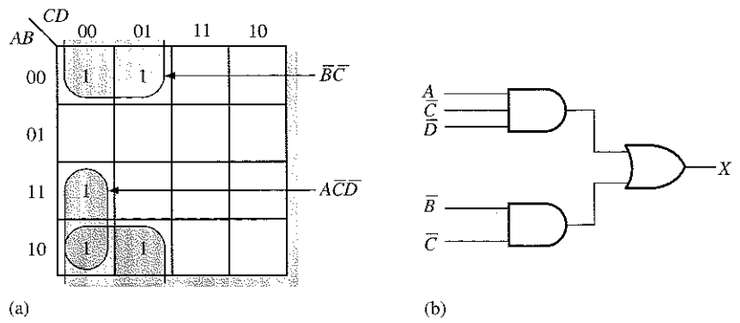


FIGURA 5.17

Problema relacionado. Desarrollar el producto de sumas equivalente del circuito de la Figura 5.17(b).

REVISIÓN DE LA SECCIÓN 5.2

1. Implementar las siguientes expresiones booleanas:
   (a) $X = ABC + AB + AC$ **(b)** $X = AB\,(C + DE)$
2. Desarrollar un circuito lógico que genere un 1 a su salida sólo cuando las tres entradas sean 1, o cuando las tres entradas sean 0.
3. Simplificar los circuitos de la cuestión 1 utilizando la suma de productos mínima.

5.3 ■ LA PROPIEDAD UNIVERSAL DE LAS PUERTAS NAND Y NOR

Hasta este momento se han estudiado los circuitos combinacionales que se implementan con puertas AND, puertas OR e inversores. En esta sección, se va a tratar la propiedad universal de la puerta NAND y de la puerta NOR. La universalidad de la puerta NAND significa que puede utilizarse como un inversor y que pueden emplearse combinaciones de la puerta NAND para implementar las operaciones AND, OR y NOR. Del mismo modo, la puerta NOR se puede utilizar para implementar el inversor y las operaciones AND, OR y NAND.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Utilizar las puertas NAND para implementar el inversor, la puerta AND, la puerta OR y la puerta NOR. ■ Utilizar las puertas NOR para implementar el inversor, la puerta AND, la puerta OR y la puerta NAND.

La puerta NAND como elemento lógico universal

La puerta NAND es una **puerta universal** porque puede utilizarse para generar la operación NOT, la operación AND, la operación OR y la operación NOR. Se puede obtener un inversor a partir de una puerta NAND conectando juntas todas las entradas, dando lugar a una única entrada, como se muestra en la Figura 5.18(a) con una puerta de 2 entradas. La operación AND se puede generar utilizando sólo puertas NAND, como muestra la Figura 5.18(b). La operación OR se puede obtener con varias puertas NAND, como ilustra la parte (c). Por último, la operación NOR se obtiene como se indica en el parte (d) de la figura.

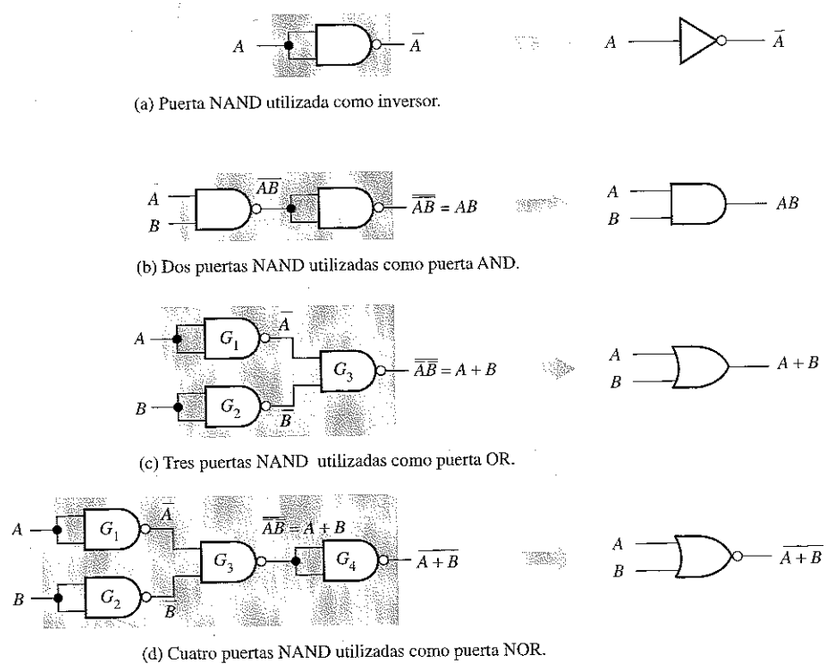


FIGURA 5.18
Aplicación universal de las puertas NAND.

En la Figura 5.18(b) se usa una puerta NAND para invertir (complementar) la salida de la otra puerta NAND, para realizar la operación AND, tal como indica la siguiente ecuación:

$$X = \overline{\overline{AB}} = AB$$

En la Figura 5.18(c), las puertas NAND, G_1 y G_2, se usan para invertir las dos variables de entrada, y después se aplican a la puerta NAND G_3. La salida OR se obtiene aplicando el teorema de DeMorgan de la siguiente forma:

$$X = \overline{\overline{A}\,\overline{B}} = A + B$$

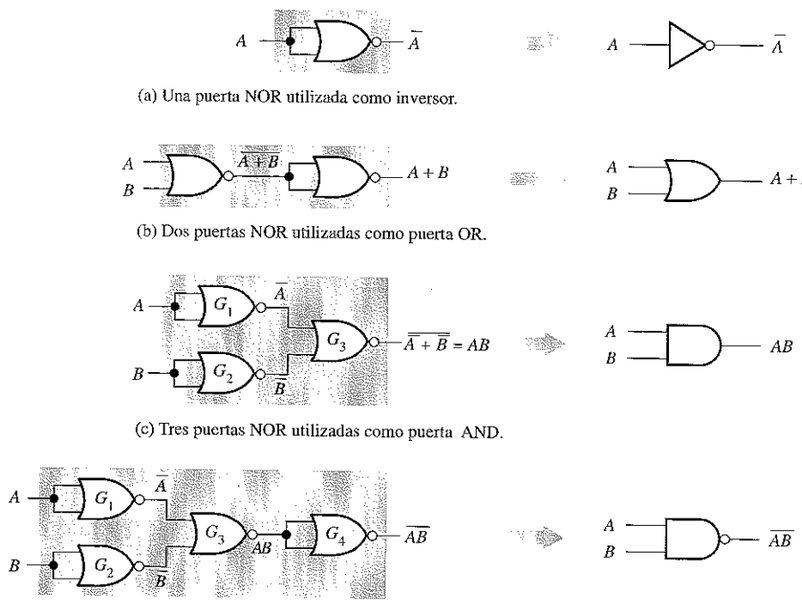
La Figura 5.18(d) utiliza la puerta NAND G_4 como inversor para generar la operación NOR, $\overline{A + B}$.

La puerta NOR como un elemento lógico universal

Al igual que la puerta NAND, la puerta NOR puede usarse para generar las operaciones NOT, AND, OR y NAND. Un circuito NOT o inversor puede obtenerse a partir de una puerta NOR conectando todas las entradas, para lograr una única entrada, como muestra la Figura 5.19(a) para 2 entradas. La puerta OR puede producirse a partir de puertas NOR, como ilustra la Figura 5.19(b). La puerta AND puede obtenerse usando puertas NOR tal y como muestra la Figura 5.19(c). En este caso, las puertas NOR, G_1 y G_2, se usan como inversores, y la salida se consigue aplicando el teorema de DeMorgan del siguiente modo:

$$X = \overline{\overline{A} + \overline{B}} = AB$$

La Figura 5.19(d) muestra cómo se disponen las puertas NOR para generar la operación NAND.



(a) Una puerta NOR utilizada como inversor.

(b) Dos puertas NOR utilizadas como puerta OR.

(c) Tres puertas NOR utilizadas como puerta AND.

(d) Cuatro puertas NOR utilizadas como puerta NAND.

FIGURA 5.19
Aplicación universal de las puertas NOR.

REVISIÓN DE LA SECCIÓN 5.3

1. Utilizar puertas NAND para implementar las expresiones:
   (a) $X = \overline{A} + B$ (b) $X = A\overline{B}$
2. Utilizar puertas NOR para implementar las expresiones:
   (a) $X = \overline{A} + B$ (b) $X = A\overline{B}$

5.4 ■ LÓGICA COMBINACIONAL UTILIZANDO LAS PUERTAS NAND Y NOR

En esta sección, se verá cómo se usan las puertas NAND y NOR para implementar una función lógica. Recuérdese, del Capítulo 3, que la puerta NAND tiene una operación equivalente denominada negativa-OR, y que la puerta NOR tiene una operación equivalente denominada negativa-AND. Veremos cómo el uso de los símbolos adecuados para representar las operaciones equivalentes hace la "lectura" del diagrama lógico más fácil.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Utilizar puertas NAND para implementar una función lógica. ■ Utilizar puertas NOR para implementar una función lógica. ■ Utilizar el símbolo apropiado en un diagrama lógico.

Circuito lógico NAND

Como ya se ha dicho, una puerta NAND puede expresarse como una función NAND o una función negativa-OR, ya que por el teorema de DeMorgan:

$$\overline{AB} = \overline{A} + \overline{B}$$

NAND — Negativa–OR

Considerando el circuito lógico de la Figura 5.20, la expresión de salida se desarrolla según los pasos siguientes:

$$\begin{aligned} X &= \overline{(\overline{AB})(\overline{CD})} \\ &= \overline{(\overline{A} + \overline{B})(\overline{C} + \overline{D})} \\ &= (\overline{\overline{A} + \overline{B}}) + (\overline{\overline{C} + \overline{D}}) \\ &= \overline{\overline{A}}\,\overline{\overline{B}} + \overline{\overline{C}}\,\overline{\overline{D}} \\ &= AB + CD \end{aligned}$$

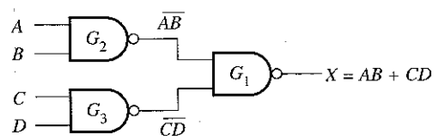
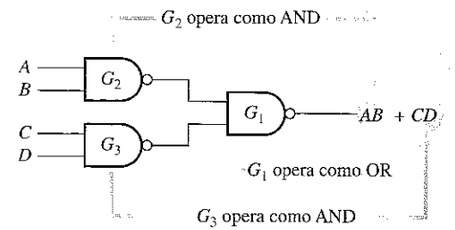
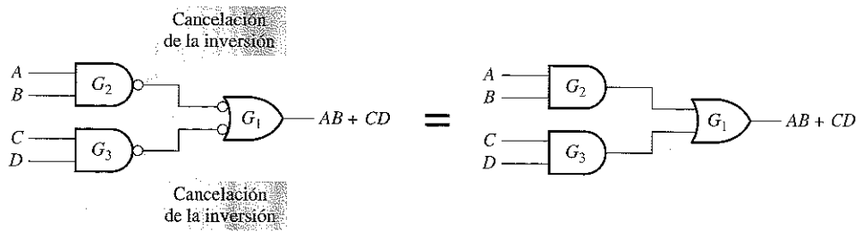


FIGURA 5.20
Circuito lógico NAND para X= AB + CD.

Obsérvese que en el último paso del desarrollo de la expresión de salida, $AB + CD$ corresponde al formato de dos términos que se multiplican (AND) y luego se suman (OR). Esta expresión muestra que las puertas G_2 y G_3 de la Figura 5.20 actúan como puertas AND, y la puerta G_1 actúa como puerta OR, como ilustra la Figura 5.21(a). En la parte (b) de esta figura se presenta este circuito con los símbolos NAND para las puertas G_2 y G_3, y un símbolo de la puerta negativa-OR para la puerta G_1.



(a) Diagrama lógico NAND original que muestra la operación de la puerta correspondiente a la expresión de salida.



(b) Diagrama lógico equivalente NAND/Negativa-OR. (c) Equivalente AND-OR.

FIGURA 5.21
Desarrollo del equivalente AND-OR del circuito de la Figura 5.20.

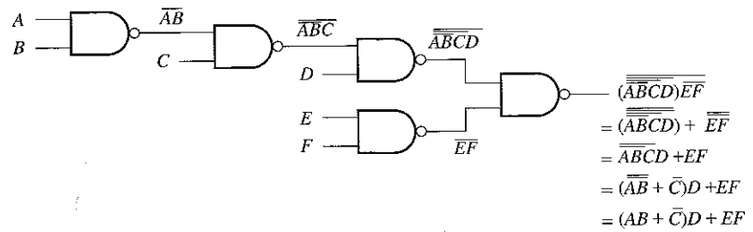
Fíjense en la Figura 5.21(b) en las conexiones círculo-círculo entre las salidas de las puertas G_2 y G_3, y las entradas de G_1. *Puesto que un círculo indica una inversión, dos círculos conectados representan una doble inversión y, por tanto, se cancelan entre sí.* Esta cancelación de inversión se ha podido ver en el desarrollo anterior para la expresión de salida $AB + CD$, y se indica por la ausencia, en la misma, de términos con una barra encima. Luego el circuito de la Figura 5.21(b) es *efectivamente* un circuito AND-OR, como se muestra en la Figura 5.21(c).

Diagrama lógico NAND

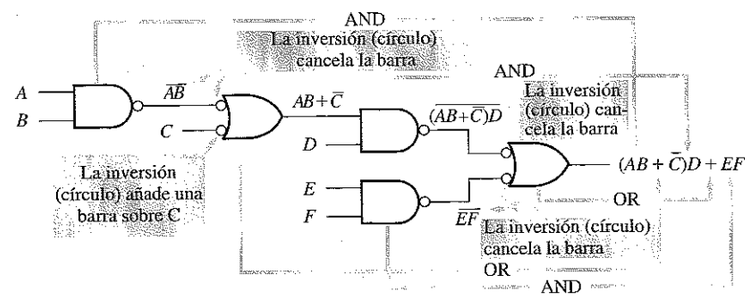
Todos los diagramas lógicos que utilizan puertas NAND deberían dibujarse utilizando el símbolo NAND o el símbolo equivalente negativa-OR para representar cada puerta, con el fin de reflejar la operación de la puerta dentro del circuito lógico. Los símbolos NAND y negativa-OR se denominan **símbolos duales**.

Si se representa la puerta de salida mediante un símbolo negativa-OR, se utilizarán entonces símbolos NAND para las puertas del nivel anterior, y se irán alternando los símbolos para los sucesivos niveles según nos alejamos de la salida. Siempre que se empleen los símbolos de este modo, la conexión entre la salida de una puerta y la entrada de otra será círculo-círculo (salida invertida-entrada invertida) o no círculo-no círculo. En un diagrama lógico, una salida con círculo no debería conectarse a una entrada sin círculo y viceversa.

La Figura 5.22 ilustra el procedimiento de utilización de los símbolos duales adecuados para un circuito NAND con varios niveles de puertas. Aunque es correcto utilizar siempre símbolos NAND, como muestra la Figura 5.22(a), el diagrama de la parte (b) es más fácil de "leer" y es preferible. La forma de las puertas indica cómo aparecerán sus entradas en la ecuación de salida y, por tanto, cómo funciona la puerta dentro del circuito lógico. Cuando se usa el símbolo NAND, las entradas aparecen multiplicadas (AND) en la expresión de salida, y cuando se usa el símbolo negativa-OR las entradas aparecen sumadas (OR), como ilustra la Figura 5.22(b). Puede verse que en el diagrama de símbolos duales de la parte (b) de la figura es mucho más fácil determinar directamente la expresión de salida. Esto se debe a que *cada puerta indica las relaciones de sus variables de entrada, tal y como aparecen en la expresión de salida.*



(a) Para llegar a la expresión de salida final son necesarios varios pasos que aplican el álgebra booleana.



(b) La expresión de salida se obtiene directamente a partir de la función de cada puerta del diagrama.

FIGURA 5.22
Ejemplo de utilización de los símbolos duales apropiados en un diagrama lógico NAND.

EJEMPLO 5.7 Volver a dibujar el diagrama lógico y desarrollar la expresión de salida para el circuito de la Figura 5.23, utilizando los símbolos duales adecuados.

Solución. En la Figura 5-24 se presenta la red de la Figura 5.23 dibujada con los símbolos de la puerta negativa-OR equivalentes. La expresión de X obtenida directamente a partir de la operación lógica indicada de cada puerta es: $X = (\overline{A} + \overline{B})C + (\overline{D} + \overline{E})F$.

FIGURA 5.23

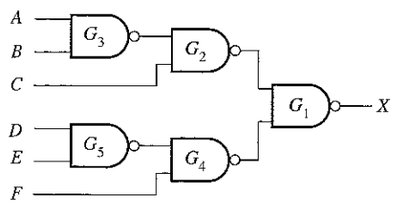
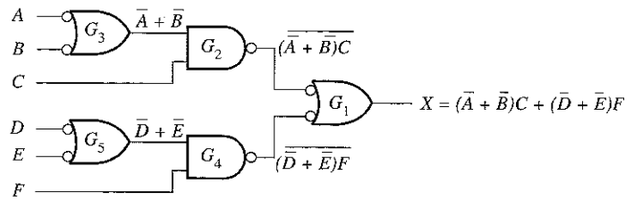


FIGURA 5.24



Problema relacionado. Obtener la expresión de salida a partir de la Figura 5.23, y demostrar que es equivalente a la expresión obtenida como solución.

EJEMPLO 5.8 Implementar las expresiones siguientes mediante lógica NAND:

(a) $ABC + DE$ **(b)** $ABC + \overline{D} + \overline{E}$

Solución. Véase Figura 5.25.

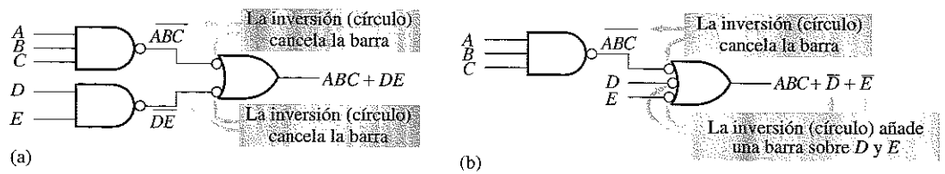


FIGURA 5.25

Problema relacionado. Convertir los circuitos NAND de las Figuras 5.25(a) y (b) a su equivalente lógico AND-OR.

Lógica NOR

Una puerta NOR puede funcionar como NOR o como negativa-AND, como demuestra el teorema de DeMorgan:

$$\overline{A + B} = \overline{A}\,\overline{B}$$

NOR — Negativa–AND

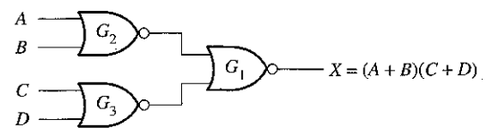


FIGURA 5.26
Diagrama lógico NOR para X = (A + B) (C + D).

Consideremos el diagrama lógico NOR de la Figura 5.26. La expresión de salida se desarrolla del siguiente modo:

$$X = \overline{\overline{A} + \overline{B} + \overline{C} + \overline{D}} = \overline{(\overline{A + B})}\,\overline{(\overline{C + D})} = (A + B)(C + D)$$

Obsérvese que la expresión $(A + B)(C + D)$ está formada por dos términos a los que primero se les aplica la operación OR y luego la operación AND. Esto implica que las puertas G_2 y G_3 operan como puertas OR, y la puerta G_1 como puerta AND, como muestra la Figura 5.27(a). Este circuito se ha dibujado de nuevo en la parte (b) de la figura con un símbolo negativa-AND para la puerta G_1.

Como en el diagrama lógico NAND, el propósito de utilizar los símbolos duales es hacer más fácil la lectura y el análisis del diagrama, lo que se ilustra en el circuito lógico NOR de la Figura 5.28. Cuando el circuito de la parte (a) se redibuja con símbolos duales, se obtiene el circuito de la parte (b) de la figura; obsérvese que todas las conexiones de salida-entrada son invertida-invertida (círculo-círculo) o no invertida-no invertida (no círculo-no círculo). De nuevo, puede comprobarse que la forma de cada puerta indica el tipo de término (AND u OR) que producirá en la expresión de salida, lo que hace que sea más fácil determinar la expresión de salida y más fácil también analizar el diagrama lógico.

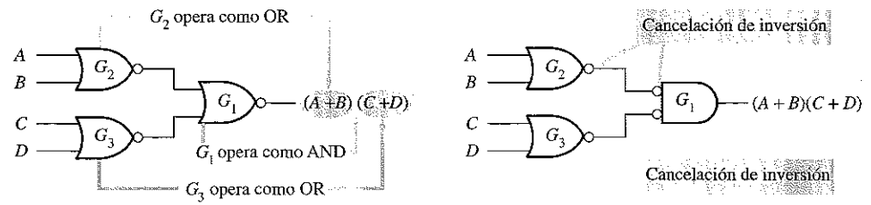
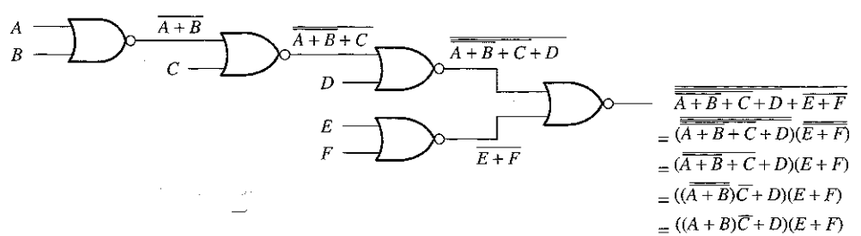


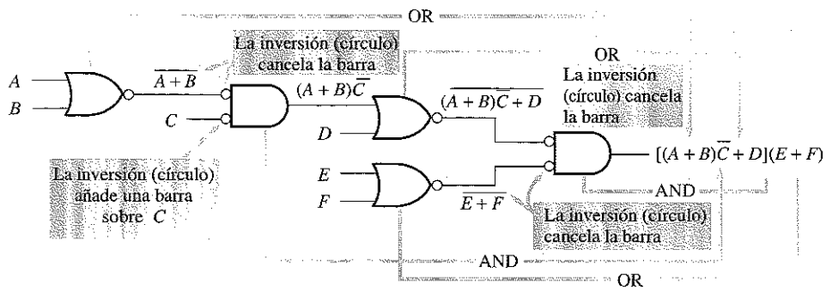
FIGURA 5.27

FIGURA 5.28
Ejemplo de utilización de los símbolos duales apropiados en un diagrama lógico NOR.



(a) La expresión de salida final se obtiene después de aplicar varios pasos del álgebra booleana.

FIGURA 5.28
(Continuación)



(b) La expresión de salida puede obtenerse directamente a partir de la función de cada puerta del diagrama.

EJEMPLO 5.9 Utilizando los símbolos duales apropiados, dibujar de nuevo el diagrama lógico y desarrollar la expresión de salida para el circuito de la Figura 5.29.

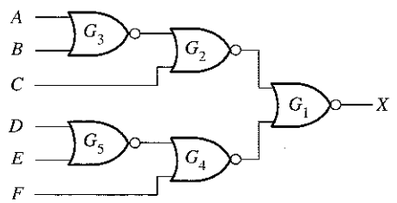


FIGURA 5.29

Solución. En la Figura 5.30 se muestra el nuevo diagrama lógico utilizando el símbolo de la puerta negativa-AND equivalente. La expresión X se obtiene directamente a partir de la función lógica de cada puerta.

$$X = (\overline{A}\,\overline{B} + C)(\overline{D}\,\overline{E} + F)$$

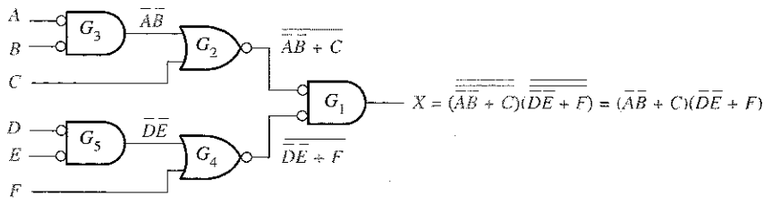


FIGURA 5.30

Problema relacionado. Demostrar que la salida del circuito NOR de la Figura 5.29 es igual que la del circuito de la Figura 5.30.

REVISIÓN DE LA SECCIÓN 5.4

1. Implementar la expresión $X = \overline{(A + \overline{B} + \overline{C})DE}$ en un diagrama lógico NAND.
2. Implementar la expresión $X = \overline{A\overline{B}\overline{C} + (\overline{D} + E)}$ en un diagrama lógico NOR.

5.5 ■ FUNCIONAMIENTO CON TRENES DE IMPULSOS

En esta sección se examinan varios ejemplos de circuitos lógicos combinacionales con entradas que son trenes de impulsos. Hay que tener en mente que la operación lógica de las puertas es la misma para impulsos que para niveles continuos de entrada. En cualquier instante determinado, la salida de un circuito lógico depende de sus entradas en ese instante, por lo que son de importancia capital las variaciones con el tiempo de las entradas.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Analizar cualquier circuito lógico combinacional con trenes de impulsos como entradas. ■ Desarrollar el diagrama de tiempos para cualquier circuito lógico combinacional de acuerdo con las entradas especificadas.

La operación lógica que realiza una puerta es la misma independientemente de que a sus entradas se apliquen impulsos o niveles constantes. La naturaleza de las entradas (impulsos o niveles constantes) no altera la tabla de verdad de un circuito. Los ejemplos de esta sección ilustran el análisis de circuitos lógicos combinacionales con impulsos de entrada.

A continuación, se presenta un repaso de la operación lógica que realiza cada puerta, con el fin de analizar los circuitos combinacionales con trenes de impulsos en sus entradas.

1. La salida de una puerta AND es un nivel ALTO (H) sólo cuando todas las entradas están a nivel ALTO (H) en el mismo instante.
2. La salida de una puerta OR es un nivel ALTO (H) siempre que al menos una de sus entradas esté a nivel ALTO (H).
3. La salida de una puerta NAND es un nivel BAJO (L) sólo cuando todas las entradas están a nivel ALTO (H) en el mismo instante.
4. La salida de una puerta NOR es un nivel BAJO (L) siempre que al menos una de las entradas esté a nivel ALTO (H).

EJEMPLO 5.10 Determinar la forma de onda de salida X para el circuito de la Figura 5.31, cuando se aplican las entradas A, B y C que se indican.

Solución. La expresión de salida, $\overline{AB + AC}$, indica que la salida X es un nivel BAJO cuando A y B están a nivel ALTO, o cuando A y C están a nivel ALTO, o cuando todas las entradas están a nivel ALTO. En la Figura 5.31 se muestra el diagrama de tiempos correspondiente a la señal de salida X. La señal de salida intermedia Y, salida de la puerta OR, se muestra en el color gris más claro.

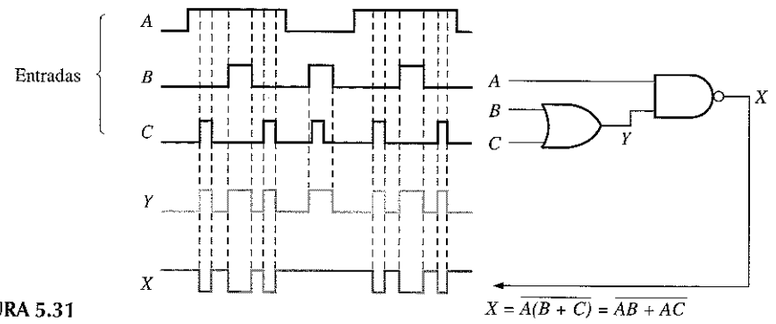


FIGURA 5.31

Problema relacionado. Determinar la forma de onda de salida si la entrada A es un nivel ALTO constante.

EJEMPLO 5.11 Dibujar el diagrama de tiempos para el circuito de la Figura 5.32, especificando las salidas de las puertas G_1, G_2 y G_3, siendo las entradas las formas de onda A y B que se indican.

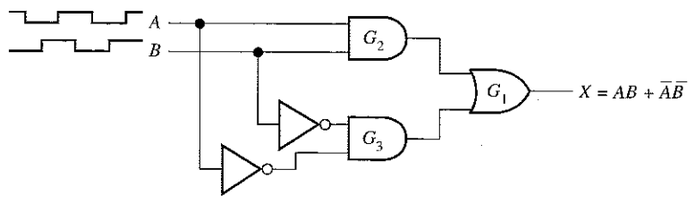


FIGURA 5.32

Solución. Cuando las dos entradas están a nivel ALTO o a nivel BAJO, la salida X se pone a nivel ALTO, como muestra la Figura 5.33. Obsérvese que se trata de un circuito NOR-exclusiva. Las salidas intermedias de las puertas G_2 y G_3 también se muestran en dicha figura.

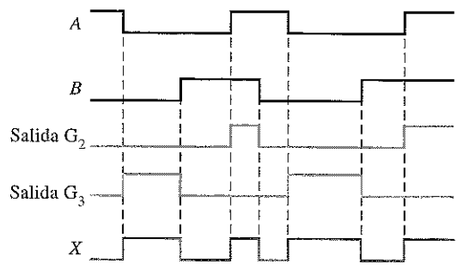


FIGURA 5.33

Problema relacionado. Determinar la salida X correspondiente al circuito de la Figura 5.32, si se invierte la entrada B.

EJEMPLO 5.12 Determinar la forma de onda de salida X para el circuito lógico de la Figura 5.34(a), hallando en primer lugar las formas de onda intermedias en los puntos Y_1, Y_2, Y_3 e Y_4. Las formas de onda de entrada son las que se indican en la Figura 5.34(b).

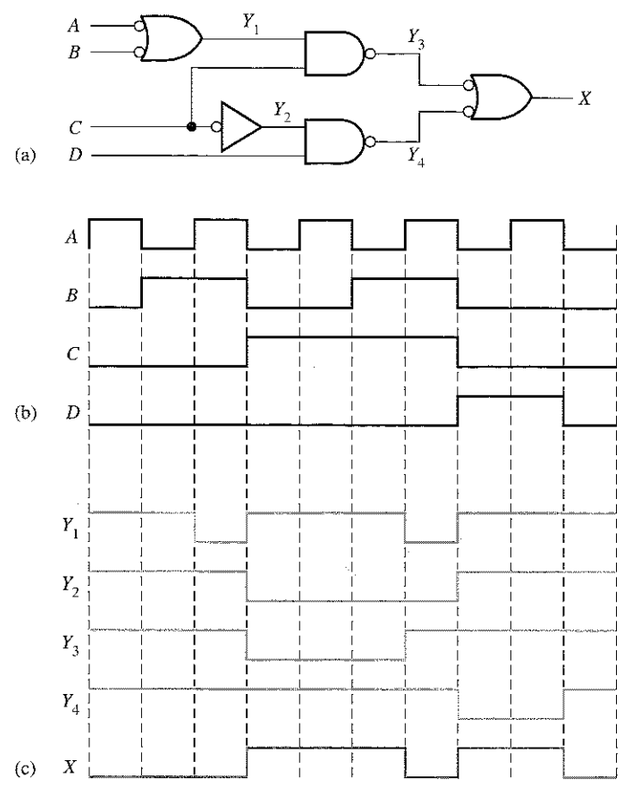


FIGURA 5.34

Solución. Todas las formas de onda intermedias y la forma de onda de salida se muestran en el diagrama de tiempos de la Figura 5.34(c).

Problema relacionado. Determinar las formas de onda Y_1, Y_2, Y_3, Y_4 y X si se invierte la entrada A.

EJEMPLO 5.13 Determinar la forma de onda de salida X para el circuito del Ejemplo 5.12, Figura 5.34(a), a partir de la expresión de salida.

Solución. En la Figura 5.35 se desarrolla la expresión de salida para el circuito especificado. La suma de productos indica que la salida es un nivel ALTO cuando A está a nivel BAJO y C está a nivel ALTO, o cuando B está a nivel BAJO y C está a nivel ALTO, o cuando C está a nivel BAJO y D está a nivel ALTO.

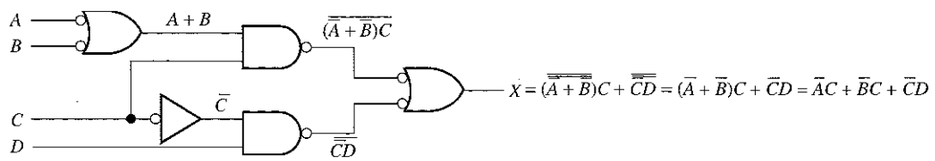


FIGURA 5.35

El resultado se muestra en la Figura 5.36, y es el mismo que se ha obtenido por el método de las señales intermedias del Ejemplo 5.12. Se indican los términos producto para cada forma de onda que dan lugar a un nivel de salida ALTO.

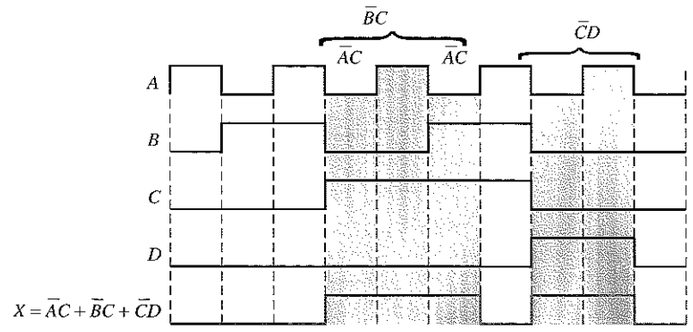


FIGURA 5.36

Problema relacionado. Repetir este ejemplo si se invierten todas las formas de onda de entrada.

REVISIÓN DE LA SECCIÓN 5.5

1. A una de las entradas de un circuito OR-exclusiva se aplica un impulso con $t_W = 50$ µs. A la otra entrada, 15 µs después de que se genere el flanco de subida del primer impulso, se aplica un impulso positivo con $t_W = 10$ µs. Dibujar la salida con relación a las entradas.
2. Los trenes de impulsos A y B de la Figura 5.31 se aplican al circuito NOR-exclusiva de la Figura 5.32. Desarrollar el diagrama de tiempos completo.

5.6 ■ LOCALIZACIÓN DE AVERÍAS

Las secciones anteriores nos han introducido en el modo de operación de los circuitos lógicos combinacionales y las relaciones entre las entradas y las salidas. Estos conocimientos son esenciales cuando hay que localizar una avería en un circuito digital, ya que se debe conocer qué niveles lógicos o señales hay que buscar en el circuito para un conjunto dado de condiciones de entrada.

En esta sección, se van a utilizar el pulsador lógico, la sonda lógica y el indicador de corriente para localizar averías en un circuito lógico cuando una salida de una puerta está conectada a varias entradas de otras puertas. También se presenta un ejemplo de métodos de análisis y seguimiento de señales utilizando un analizador lógico para localizar un fallo en un circuito lógico combinacional.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Definir nodo de circuito. ■ Utilizar una sonda lógica para encontrar un fallo en un nodo de circuito. ■ Utilizar un pulsador y una sonda para encontrar un circuito abierto en la salida de una puerta. ■ Utilizar un pulsador lógico y un indicador de corriente para encontrar un cortocircuito en la salida o la entrada de una puerta. ■ Utilizar un osciloscopio o un analizador lógico para seguir una señal en un circuito lógico combinacional.

En un circuito lógico combinacional, la salida de una puerta puede conectarse a dos o más entradas de otra puerta, como muestra la Figura 5.37. Las interconexiones se cruzan en un punto eléctrico común que se denomina **nodo**.

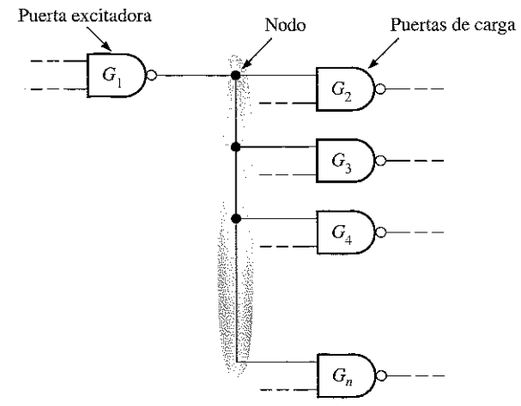


FIGURA 5.37
Ilustración de un nodo en un circuito lógico.

La puerta G_1 de la Figura 5.37 excita al nodo, y las demás puertas representan las cargas conectadas al nodo. Una puerta excitadora puede excitar a un determinado número de entradas de puertas de carga, hasta el máximo determinado por su *fan-out* específico. En esta situación, se pueden producir diversos tipos de fallos. Algunos de

estos tipos de fallos son difíciles de aislar en una puerta, ya que todas las puertas conectadas al nodo se ven afectadas. Los fallos más comunes son los siguientes:

1. *Salida en circuito abierto en la puerta excitadora.* Este fallo da lugar a pérdida de la señal en todas las puertas de carga.
2. *Entrada en circuito abierto en una puerta de carga.* Este fallo no afectará al funcionamiento de ninguna otra puerta conectada al nodo, pero hará que no se detecte señal de salida en la puerta que falla.
3. *Salida cortocircuitada de la puerta excitadora.* Este fallo puede dar lugar a que el nodo permanezca en estado BAJO (cortocircuitado a masa) o en estado ALTO (cortocircuitado a V_{CC}).
4. *Entrada cortocircuitada en una puerta de carga.* Este fallo también hace que el nodo se mantenga a nivel BAJO (cortocircuitado a masa) o en estado ALTO (cortocircuitado a V_{CC}).

Consejos Prácticos

Cuando se están localizando averías en circuitos lógicos, se debe empezar por una comprobación visual con el fin de localizar los problemas obvios. La inspección visual debería incluir los conectores además de los componentes. Los conectores externos se usan frecuentemente para llevar a una tarjeta de circuito las señales, la alimentación y la masa. Las superficies de contacto de los conectores deben estar limpias y deben tener una buena fijación mecánica. Un conector sucio puede producir un fallo intermitente o completo del circuito. Los conectores se pueden limpiar con un borrador de lápiz normal y un bastoncillo humedecido en alcohol. También se deberían comprobar todos los conectores para localizar los pines que no estén bien ajustados.

Localización de los fallos más comunes

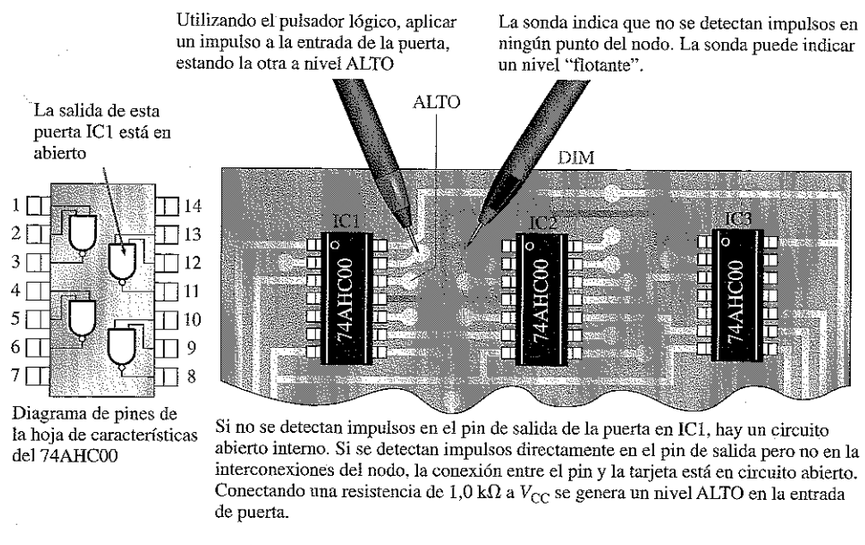
Salida en circuito abierto en la puerta excitadora. En este caso no se detectan impulsos en el nodo. Con el circuito alimentado, un nodo en circuito abierto dará lugar, normalmente, a un nivel "flotante", lo que puede indicarse mediante una luz débil en la sonda lógica, como se ilustra en la Figura 5.38.

Entrada en circuito abierto en una puerta de carga. Si la salida de la puerta excitadora no está en circuito abierto, entonces hay que probar si la entrada de una puerta de carga está en circuito abierto. Se aplica la punta del pulsador lógico al nodo estando todos los niveles de entrada en estado ALTO. Luego, con la sonda lógica se comprueba la salida de cada una de las puertas, como se indica en la Figura 5.39. Si una de las entradas conectadas al nodo está en circuito abierto, no se detectarán impulsos en la salida de la misma.

Salida en cortocircuito a masa de la puerta excitadora. Este fallo puede hacer que el nodo permanezca a nivel BAJO, como se ha dicho anteriormente. Una rápida comprobación con el pulsador lógico y la sonda lógica lo detectarán, como se muestra en la Figura 5.40(a). Un cortocircuito a masa en la salida de la puerta excitadora o en la entrada de cualquier puerta de carga dará lugar a este síntoma, por lo

que deben hacerse más comprobaciones para aislar el cortocircuito en una puerta en concreto.

FIGURA 5.38
Salida en circuito abierto en la puerta.

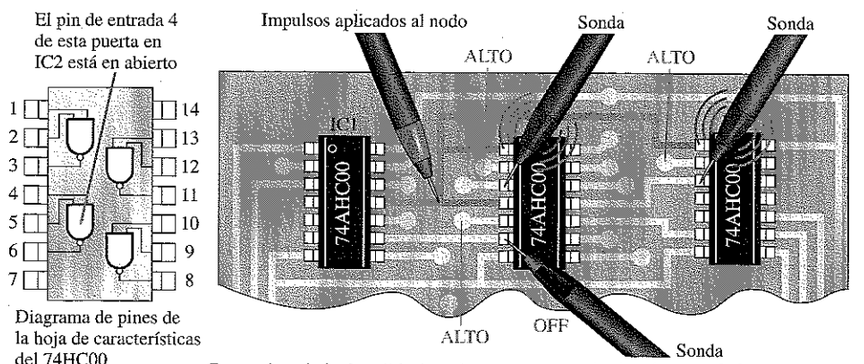


Si no se detectan impulsos en el pin de salida de la puerta en IC1, hay un circuito abierto interno. Si se detectan impulsos directamente en el pin de salida pero no en la interconexiones del nodo, la conexión entre el pin y la tarjeta está en circuito abierto. Conectando una resistencia de 1,0 kΩ a V_{CC} se genera un nivel ALTO en la entrada de puerta.

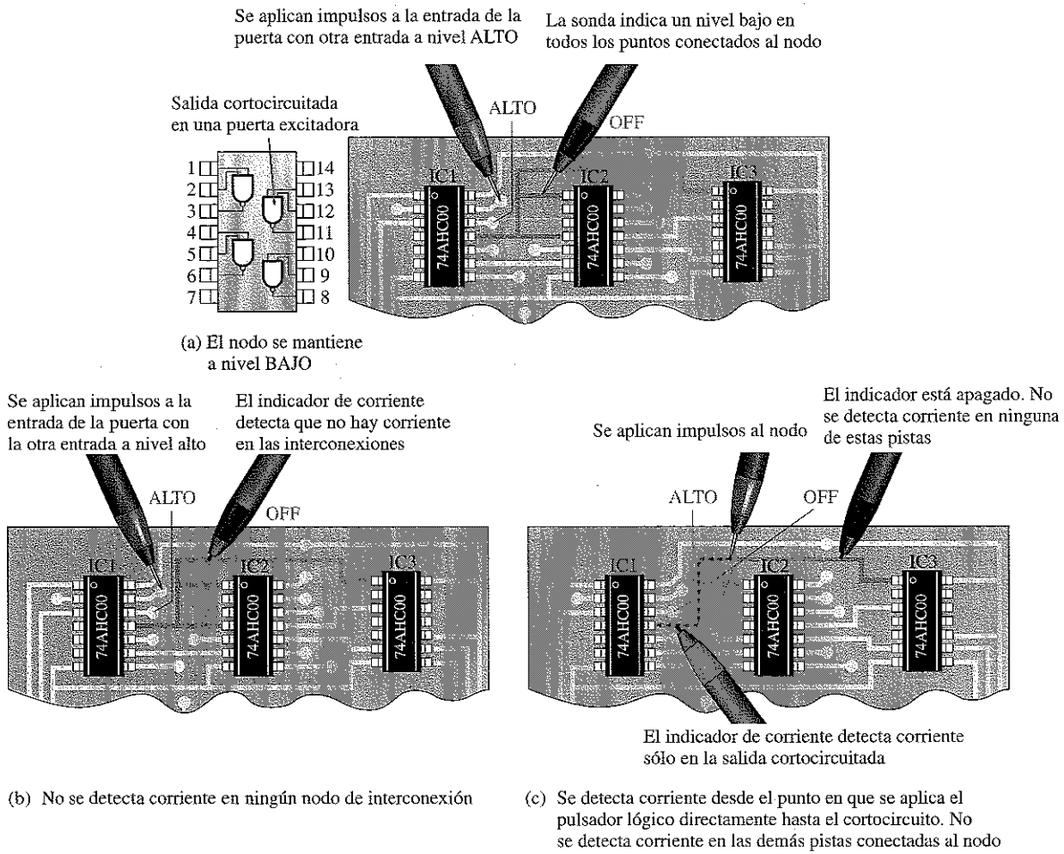
Si la salida de la puerta excitadora está internamente cortocircuitada a masa, entonces no se detectará corriente en ninguna de las conexiones al nodo. Por tanto, el indicador de corriente no detectará corriente cuando está conectada la alimentación, tal y como ilustra la Figura 5.40(b)

Como muestra la Figura 5.40(c), para verificar una salida cortocircuitada se pueden utilizar un indicador de corriente y un pulsador lógico cuando la alimentación está desconectada. Cuando se aplican los impulsos de corriente con el pulsador lógico al nodo, toda la corriente se derivará por la salida cortocircuitada, y no circulará nada por las entradas de las puertas de carga.

FIGURA 5.39
Entrada en circuito abierto de una puerta de carga.



Comprobar el pin de salida de cada puerta conectada al nodo, estando las restantes entradas de la puerta a nivel ALTO. Si no se detectan impulsos en una salida quiere decir que una entrada o una salida de puerta está en circuito abierto. Conectando una resistencia de 1,0 kΩ a V_{CC} se genera un nivel ALTO en las entradas de la puerta.



(a) El nodo se mantiene a nivel BAJO

(b) No se detecta corriente en ningún nodo de interconexión

(c) Se detecta corriente desde el punto en que se aplica el pulsador lógico directamente hasta el cortocircuito. No se detecta corriente en las demás pistas conectadas al nodo

FIGURA 5.40
Salida cortocircuitada en una puerta excitadora.

NOTAS INFORMÁTICAS
El probar los circuitos de más reciente fabricación está siendo cada vez más difícil debido al incremento de la complejidad y al reducido tamaño de los componentes que sólo permiten un acceso restringido a los pines. Un método que está comenzando a popularizarse para probar circuitos de alta densidad se basa en probar mediante una computadora una tarjeta lógica que utilice circuitos integrados que contienen lógica predefinida adicional, denominada lógica de exploración de contorno *(boundary scan logic)*. El concepto consiste en añadir puntos de prueba (celdas de exploración de contorno) dentro del CI en los puntos de E/S de un circuito lógico. Cada dispositivo tiene pines adicionales que permiten emplear la lógica de exploración de contorno. Los estados del dispositivo se pueden leer y controlar a través de una señal de entrada serie a una computadora, que controla las pruebas y puede aislar los fallos en el pin en el que se produce el error.

Entrada cortocircuitada a masa en una puerta de carga. Si una de las puertas de carga está internamente cortocircuitada a masa, el nodo se mantendrá a nivel BAJO. De nuevo, nos encontramos en el caso de una salida cortocircuitada, luego se pueden emplear el pulsador lógico y el indicador de corriente para aislar la puerta que falla.

Cuando se aplican impulsos al nodo estando desconectada la alimentación, toda la corriente se derivará a la entrada cortocircuitada, y siguiendo las pistas con el indicador de corriente se llegará a la entrada cortocircuitada, tal y como muestra la Figura 5.41.

Análisis y seguimiento de señales

Aunque los métodos de aislamiento de cortocircuitos y circuitos abiertos en un nodo son muy útiles en ocasiones, una técnica más general para la localización de fallos es el **seguimiento de señales**, la cual tiene un gran valor para el técnico en casi todos los casos de fallo. La medida de la señal se realiza con un osciloscopio o un analizador lógico.

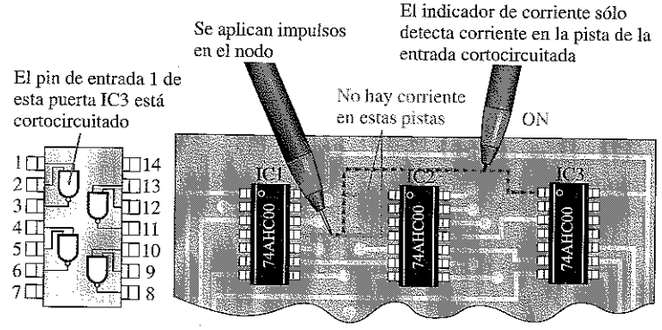


FIGURA 5.41
Entrada cortocircuitada en una puerta de carga.

Básicamente, el método de seguimiento de señales requiere que se observe la señal y su cronograma en todos los puntos accesibles del circuito lógico. Se puede comenzar por las entradas y, a partir del análisis del diagrama de tiempos de la señal en cada punto, determinar cuál es el primer punto en que la señal es incorrecta. Normalmente, con este procedimiento se puede aislar el fallo en una puerta específica. También se puede usar el método de comenzar por la salida y continuar hacia atrás hasta las entradas.

El procedimiento general del seguimiento de señales comenzando por las entradas se describe a continuación:

- Dentro del sistema, definir la sección del circuito lógico que se sospecha que está fallando.
- Comenzar en las entradas de la sección que se va a examinar. Para este estudio, suponemos que las formas de onda de entrada proceden de otras partes del sistema que son correctas.
- Para cada puerta, empezando por la entrada y yendo hacia la salida del circuito lógico, se observa la forma de onda de salida de la puerta y se compara con las formas de onda de entrada, utilizando el osciloscopio o el analizador lógico.
- Determinar si la señal de salida es correcta utilizando nuestros conocimientos sobre la operación lógica de la puerta.
- Si la salida es incorrecta, en la puerta bajo prueba puede estar el fallo. Extraiga el CI que contiene la puerta de la que se sospecha que produce el fallo, y compruébelo fuera del circuito. Si la puerta falla, reemplace el CI. Si funciona correctamente, el fallo está en la circuitería externa o en otro CI al que está conectado el que se está probando.
- Si la salida es correcta, pase a la puerta siguiente. Continúe comprobando cada puerta hasta observar una forma de onda incorrecta.

La Figura 5.42 es un ejemplo que ilustra el procedimiento general para un circuito lógico específico, en el que se siguen los pasos que a continuación se indican:

Paso 1. Observar la salida de la puerta G_1 (punto de prueba 5) respecto a sus entradas. Si es correcta, probar el inversor siguiente. Si la salida no es correcta, la puerta o sus conexiones están mal; o, si la salida está a nivel BAJO, la entrada de la puerta G_2 puede estar cortocircuitada.

Paso 2. Observar la salida del inversor (TP6) respecto a la entrada. Si es correcta, probar la puerta siguiente, G_2. Si la salida no es correcta, el inversor o sus conexiones están mal; o, si la salida está a nivel BAJO, la entrada de la puerta G_3 puede estar cortocircuitada.

Paso 3. Observar la salida de la puerta G_2 (TP7) respecto a las entradas. Si es correcta, probar la puerta siguiente, G_3. Si la salida no es correcta, el inversor o sus conexiones están mal; o, si la salida está a nivel BAJO, la entrada de la puerta G_4 puede estar cortocircuitada.

Paso 4. Observar la salida de la puerta G_3 (TP8) respecto a sus entradas. Si es correcta, probar la puerta G_4. Si la salida no es correcta, la puerta o sus conexiones están mal; o, si la salida está a nivel BAJO, la entrada de la puerta G_4 (TP9) puede estar cortocircuitada.

Paso 5. Observar la salida de la puerta G_4 respecto a sus entradas. Si es correcta, el circuito está bien. Si la salida no es correcta, la puerta o sus conexiones están mal.

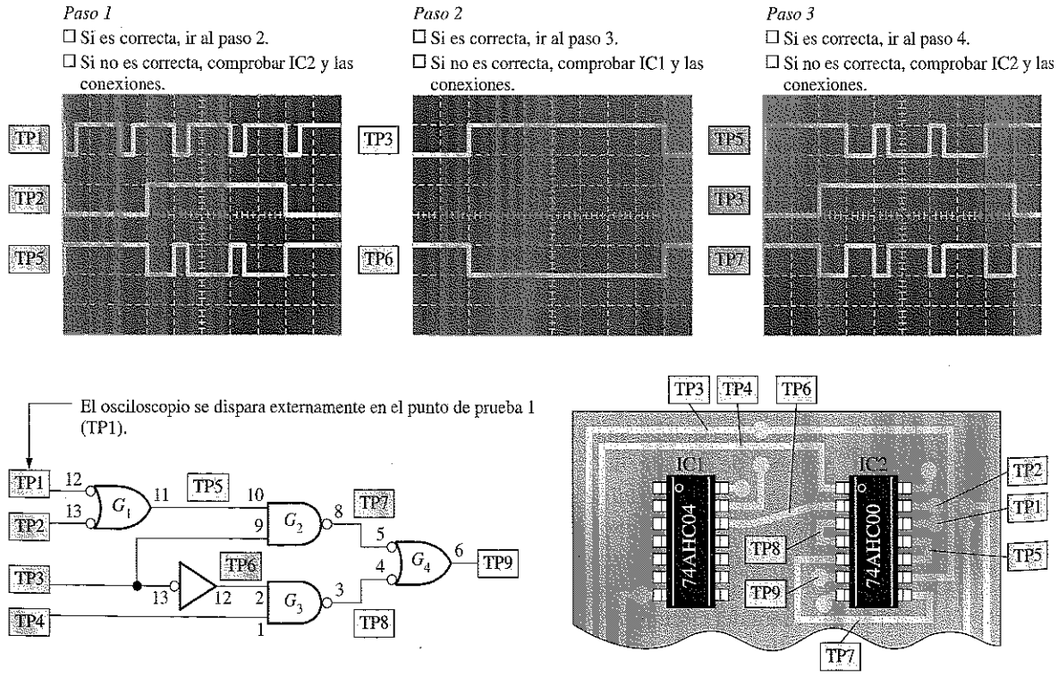


FIGURA 5.42
Ejemplo de análisis y seguimiento de señales en una parte de un sistema digital. TP indica punto de prueba (test point).

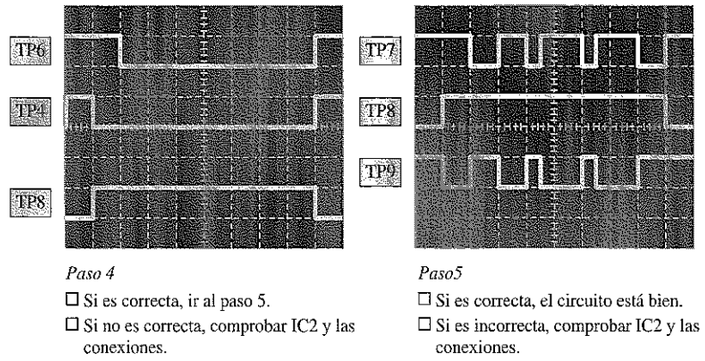


FIGURA 5.42
(Continuación).

EJEMPLO 5.14 Determinar el fallo en el circuito lógico de la Figura 5.43(a) utilizando el análisis de señales. Las formas de onda de color gris son correctas, por lo que debe observar las señales en negro de la Figura 5.43(b).

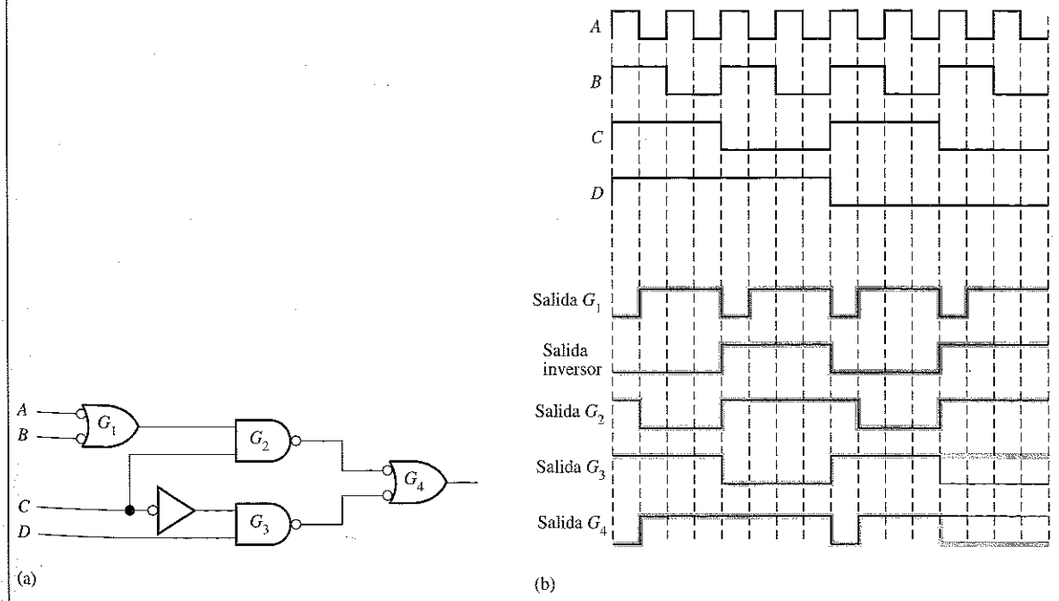


FIGURA 5.43

Solución

1. Determinar cuál es la forma de onda correcta para cada puerta. Las formas de onda correctas se muestran en color gris en la Figura 5.43(b), sobreimpresas a las formas de onda medidas realmente.

2. Comparar las formas de onda puerta por puerta hasta encontrar una señal medida que no se corresponda con la señal correcta.

En este ejemplo, todas las comprobaciones son correctas hasta llegar a la puerta G_3. La salida de ésta es incorrecta de acuerdo con las diferencias que se indican. Un análisis de la señal pone de manifiesto que, si la entrada D de la puerta G_3 está en circuito abierto y opera como un nivel ALTO, se obtendría la forma de onda de salida medida mostrada en color gris. Obsérvese que la salida de G_4 es correcta para las entradas medidas, aunque la entrada procedente de G_3 es incorrecta.

Reemplazar el circuito integrado que contiene a G_3, y probar de nuevo el funcionamiento del circuito.

Problema relacionado. Para las entradas de la Figura 5.43(b), determinar la forma de onda de salida del circuito lógico (salida de G_4), si la salida del inversor está en circuito abierto.

Consejos Prácticos

Como ya sabemos, probar y localizar los fallos en circuitos lógicos frecuentemente requiere observar y comparar dos formas de onda digitales de forma simultánea, tal como una entrada y la salida de una puerta, en un osciloscopio de doble canal. Para las formas de onda digitales, el osciloscopio siempre debería configurarse con acoplamiento DC en cada entrada de canal, para evitar los "desplazamientos" del nivel de la señal. Se deberá determinar dónde se encuentra el nivel de 0 V en la pantalla para ambos canales.

Para comparar la temporización de las formas de onda, el osciloscopio debería dispararse sólo desde un canal (no utilizar el disparo en modo vertical o compuesto). El canal seleccionado para disparo debería ser siempre aquel que tenga la frecuencia más baja, cuando sea posible.

REVISIÓN DE LA SECCIÓN 5.6

1. Enumerar cuatro tipos de fallos comunes en las puertas lógicas.
2. Una entrada de una puerta NOR está externamente cortocircuitada a $+V_{CC}$. ¿Cómo afecta esta condición al funcionamiento de la puerta?
3. Determinar la salida de la puerta G_4 de la Figura 5.43(a), si se aplican las entradas de la Figura 5.43(b), para los fallos siguientes:
   (a) Una entrada de G_1 cortocircuitada a masa.
   (b) La entrada del inversor cortocircuitada a masa.
   (c) Una salida en circuito abierto en G_3.

5.7 ■ APLICACIÓN A LOS SISTEMAS DIGITALES

En esta aplicación, se introduce un sistema de control digital para una parte de una operación de tratamiento de madera de una fábrica de muebles. Este sistema controla cuatro motores que ponen en marcha una cinta transportadora, su bomba de lubricación y dos sierras.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir el funcionamiento del sistema. ■ Diseñar la lógica de control para una parte del sistema. ■ Localizar los fallos en la tarjeta del sistema. ■ Desarrollar un procedimiento de pruebas para la lógica de control.

Funcionamiento básico del sistema

Como puede verse en el diagrama de bloques de la Figura 5.44, el sistema utiliza cuatro interruptores manuales on/off, un bloque de lógica de control y una interfaz de excitación de los motores para controlar el motor de la bomba de lubricación de la cinta transportadora, el motor de la cinta transportadora, el motor de la sierra de corte y el motor de la sierra de cinta. Esta aplicación está enfocada a la fase de la lógica de control del sistema.

Requerimientos de operación. La entrada del interruptor S_1 controla el motor de la bomba de lubricación (salida M_1). La entrada del interruptor S_2 controla el motor de la cinta transportadora (salida M_2). La entrada del interruptor S_3 controla el motor de la sierra de cinta (salida M_3). La entrada del interruptor S_4 controla el motor de la sierra de corte (salida M_4).

El motor que controla la lubricación de la cinta transportadora debe funcionar ($M_1 = 1$) cuando la cinta transportadora está funcionando. El motor que controla a la cinta debe funcionar ($M_2 = 1$) sólo cuando los interruptores 1 y 2 están activados ($S_1 = 1$ y $S_2 = 1$). El motor de la sierra de cinta funciona ($M_3 = 1$) cuando el interruptor 3 está activado ($S_3 = 1$), y el motor de la sierra de corte funciona ($M_4 = 1$) cuando el interruptor 4 está activado ($S_4 = 1$). Los motores de las sierras no requieren lubricación, pero nunca deben funcionar al mismo tiempo. Si los interruptores 3 y 4 se activan al mismo tiempo, el sistema debe pararse por completo, incluyendo los motores de la cinta transportadora y de lubricación. Tampoco pueden funcionar al tiempo la cinta transportadora y la sierra de corte. La lógica de control controla los motores para evitar que se produzca cualquier condición no permitida debido a la manipulación incorrecta de los interruptores.

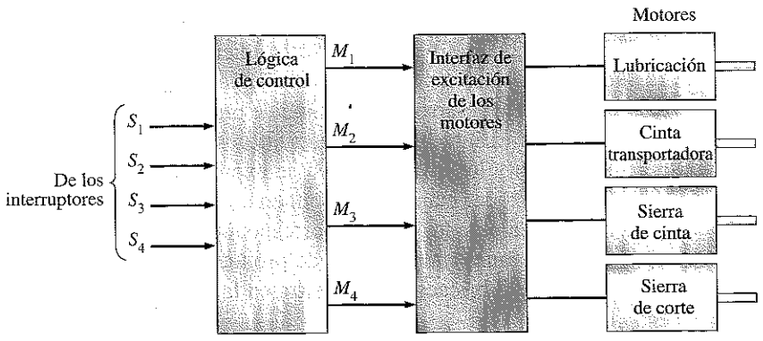


FIGURA 5.44
Diagrama de bloques básico del sistema de control.

Tabla de verdad de la lógica de control

La Tabla 5.6 es la tabla de verdad que muestra los estados de los interruptores y los motores, basándose en las condiciones descritas. Un 1 indica un estado ON (activación) y un 0 indica un estado OFF (desactivación). Puesto que hay cuatro interruptores, existen dieciséis posibles combinaciones ON/OFF para los interruptores y, por tanto, dieciséis posibles combinaciones para los motores. Los estados de los interruptores son las variables de entrada y los estados de los motores son las variables de salida. En este caso, las condiciones no permitidas no se tratan como "indiferentes". Se trata de condiciones de entrada de los interruptores que no deberían producirse pero, si se producen, el sistema debe desconectarse y parar todos los motores.

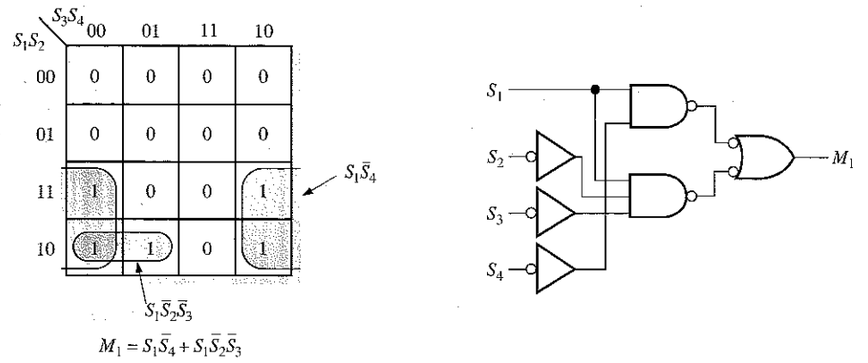
TABLA 5.6
Tabla de verdad de la lógica de control.

Entradas				Salidas				
S_1	S_2	S_3	S_4	M_1 Lub	M_2 Cinta	M_3 Banda	M_4 Corte	Comentarios
0	0	0	0	0	0	0	0	Apagar el sistema
0	0	0	1	0	0	0	1	Correcto
0	0	1	0	0	0	1	0	Correcto
0	0	1	1	0	0	0	0	No permitido (2 sierras activadas). Apagar
0	1	0	0	0	0	0	0	No permitido (cinta sin lubricante). Apagar
0	1	0	1	0	0	0	0	No permitido (cinta y corte). Apagar
0	1	1	0	0	0	0	0	No permitido (cinta sin lubricante). Apagar
0	1	1	1	0	0	0	0	No permitido (2 sierras activadas). Apagar
1	0	0	0	1	0	0	0	Correcto
1	0	0	1	1	0	0	1	Correcto
1	0	1	0	1	0	1	0	Correcto
1	0	1	1	0	0	0	0	No permitido (2 sierras). Apagar
1	1	0	0	1	1	0	0	Correcto
1	1	0	1	0	0	0	0	No permitido (cinta y corte). Apagar
1	1	1	0	1	1	1	0	Correcto
1	1	1	1	0	0	0	0	No permitido (todo activado). Apagar

Diseño de la lógica de control

Hay cuatro circuitos lógicos separados, uno para cada motor. Vamos a empezar por el diseño del circuito lógico correspondiente al motor de la bomba de lubricación (salida M_1). El primer paso es trasladar los datos de la tabla de verdad a un mapa de Karnaugh y desarrollar una suma de productos.

Las variables de los interruptores, S_1, S_2, S_3 y S_4 son las variables del mapa de Karnaugh y los estados de M_1 se expresan y agrupan como muestra la Figura 5.45 (a). Los 0s del mapa corresponden a las condiciones de los interruptores cuando el motor está parado, y los 1s corresponden a la condición de los interruptores cuando el motor funciona. La suma de productos resultante para la lógica del motor de la bomba de lubricación se muestra en la implementación NAND de la parte (b) de la figura.

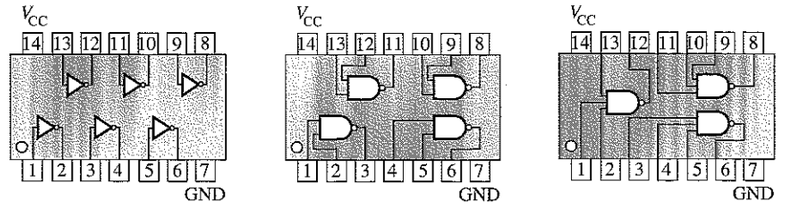


(a) (b)

FIGURA 5.45
Simplificación mediante el mapa de Karnaugh e implementación para la lógica del motor de lubricación (M1).

■ PRÁCTICAS DE SISTEMAS

Se ha diseñado la lógica del motor de lubricación y ahora le toca al lector diseñar la lógica de control de los tres motores restantes. Cuando finalice el diseño, éste debería estar de acuerdo con la lógica de la tarjeta del sistema. Todas las conexiones a la alimentación V_{CC} y a masa se encuentran en la cara posterior de la tarjeta y no se muestran, para mayor simplicidad. Las interconexiones del circuito en la parte posterior de la tarjeta pasan a través de los taladros hasta la cara de componentes. La Figura 5.46 muestra la definición de los pines para los circuitos integrados de la tarjeta.

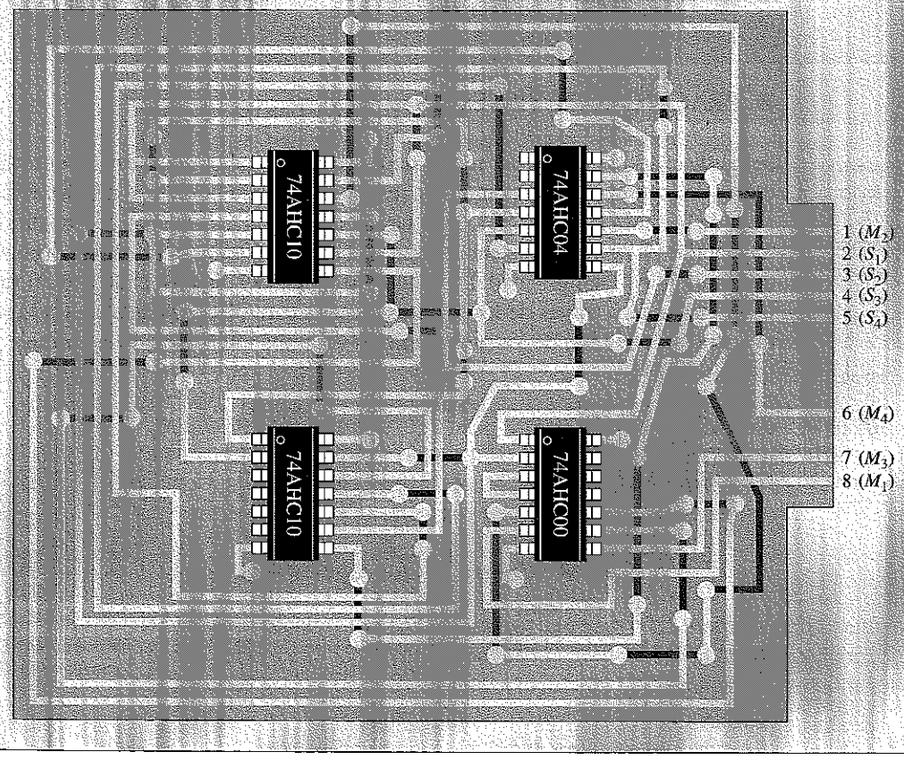


(a) Sextuple inversor 74AHC04. (b) Cuádruple puerta NAND de 2 entradas 74AHC00. (c) Triple puerta NAND de 3 entradas 74AHC10.

FIGURA 5.46
Diagrama lógico de pines de los circuitos integrados.

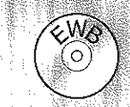
■ PRÁCTICAS DE SISTEMAS 1: Diseño

- ***Actividad 1.*** Utilizando la Tabla 5.6 y el método de mapa de Karnaugh, diseñar la lógica para controlar la cinta transportadora utilizando puertas NAND e inversores.
- ***Actividad 2.*** Diseñar la lógica para controlar el motor de la sierra de cinta.
- ***Actividad 3.*** Diseñar la lógica para controlar el motor de la sierra de corte.
- ***Actividad 4.*** Combinar la lógica de cada uno de los controles de los cuatro motores en un diagrama lógico completo. Especificar los dispositivos que se deben utilizar.
- ***Actividad 5.*** Trazar el circuito impreso y dibujar el diagrama lógico resultante con la ayuda del diagrama de pines de la Figura 5.46. La lógica que diseñe debería estar de acuerdo con la lógica de la tarjeta (a excepción de la numeración de algunos pines).
- ***Actividad 6.*** Etiquetar las entradas y las salidas de la tarjeta de circuito impreso utilizando las variables de la tabla de verdad.

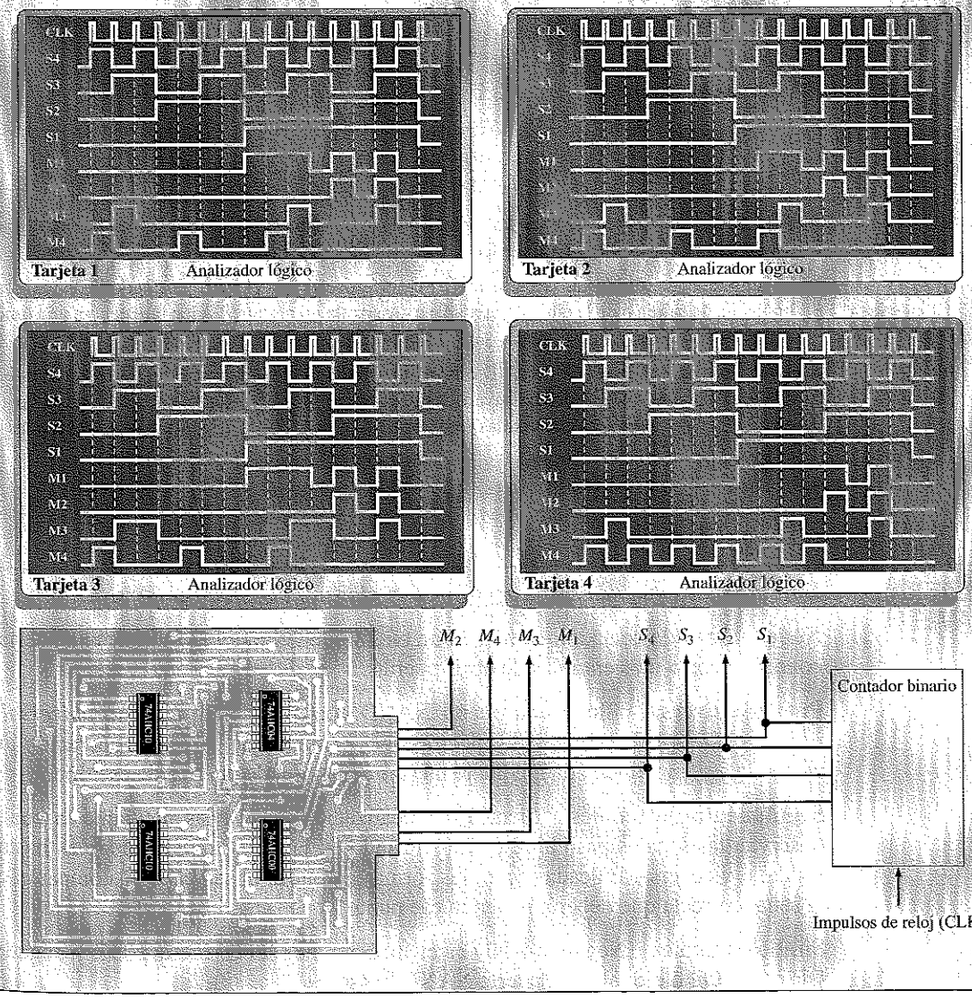


■ PRÁCTICAS DE SISTEMAS 2: Pruebas y localización de averías

- ***Actividad 1.*** Para probar la tarjeta, se debe conectar a la misma un dispositivo (contador binario) que genere una secuencia de números binarios desde 0 hasta 15, para simular las entradas de los interruptores de acuerdo con la Tabla 5.6. Se utiliza un analizador lógico para observar las formas de onda de salida. Para cada conjunto de señales de salida, determinar si la tarjeta funciona correctamente, y si no es así, especificar los posibles fallos.
- ***Actividad 2.*** Diseñar un método manual que se pueda emplear para probar la lógica de control.



La lógica de control se encuentra en el archivo SA05.EWB del CD-ROM EWB. Utilícese el generador y el analizador lógico para simular esta prueba para cada uno de los fallos.



REVISIÓN DE LA SECCIÓN 5.7

1. Establecer el objetivo de la lógica de control de esta aplicación.
2. ¿Qué ocurre si los interruptores 1 y 3 se activan a la vez, estando desactivados los interruptores 2 y 4?
3. ¿Qué ocurre si los interruptores 2 y 4 se activan a la vez, estando desactivados los interruptores 1 y 3?

En este punto puede estudiarse el Capítulo 15, "Tecnología de circuitos integrados", parcial o totalmente. El examen de este capítulo no es un prerrequisito para abordar cualquier otro tema tratado en este texto, y su omisión total o parcial no afectará al resto de los temas.

■ RESUMEN

- La lógica AND-OR genera una expresión de salida en formato suma de productos.
- La lógica AND-OR-Inversor genera una suma de productos complementada, que es realmente un producto de sumas.
- El símbolo operacional para OR-Exclusiva es ⊕. Una expresión OR-Exclusiva puede expresarse de dos formas equivalentes:

$$A\overline{B} + \overline{A}B = A \oplus B$$

- Para llevar a cabo el análisis de un circuito lógico, se debe desarrollar la expresión booleana de salida del mismo o su tabla de verdad, o ambas.
- La implementación de un circuito lógico es el proceso por el que, partiendo de la expresión booleana de salida o de la tabla de verdad, se desarrolla un circuito lógico que genera una función de salida.
- Todos los diagramas lógicos NAND o NOR deben dibujarse utilizando los apropiados símbolos duales, de modo que las salidas invertidas (con círculo) se conectan a entradas invertidas (con círculo), y las salidas no invertidas (sin círculo) se conectan a las entradas no invertidas (sin círculo).
- Cuando se conectan dos indicadores de negación (círculos), éstos se cancelan entre sí.

■ PALABRAS CLAVE

Estos términos también se encuentran en el glosario del final del libro.

Nodo Punto de conexión común en un circuito, en el que la salida de una puerta se conecta a una o más entradas de otras puertas.

Puerta universal Tanto una puerta NAND como NOR. El término *universal* se refiere a la propiedad de aquellas puertas que permiten que cualquier operación lógica pueda ser implementada mediante una combinación de puertas de ese tipo.

Seguimiento de señales Técnica de localización de averías mediante la cual se observan las señales paso a paso, comenzando en la entrada y siguiéndolas hasta la salida, o viceversa. En cada punto, las formas de ondas observadas se comparan con la señal correcta que debería haber en ese punto.

Símbolos duales Negativa-AND es el símbolo dual de la puerta NOR y negativa-OR es el símbolo dual de la puerta NAND.

■ AUTOTEST

1. La expresión de salida de un circuito AND-OR que consta de una puerta AND con las entradas *A, B, C* y *D*, y otra puerta AND con entradas *E* y *F* es:
   (a) *ABCDEF* **(b)** $A + B + C + D + E + F$
   (c) $(A + B + C + D)(E + F)$ **(d)** $ABCD + EF$
2. Un circuito lógico con una salida $X = A\overline{B}C + A\overline{C}$ consta de:
   (a) dos puertas AND y una puerta OR.
   (b) dos puertas AND, una puerta OR y dos inversores.
   (c) dos puertas OR, una puerta AND y dos inversores.
   (d) dos puertas AND, una puerta OR y un inversor.
3. Para implementar la expresión $\overline{A}BCD + A\overline{B}CD + AB\overline{C}\overline{D}$ se utiliza una puerta OR y
   (a) una puerta AND
   (b) tres puertas AND
   (c) tres puertas AND y cuatro inversores
   (d) tres puertas AND y tres inversores
4. La expresión $\overline{A}BCD + ABC\overline{D} + A\overline{B}\overline{C}D$
   (a) no se puede simplificar
   (b) se puede simplificar a $\overline{A}BC + A\overline{B}$
   (c) puede simplificarse a $ABC\overline{D} + \overline{A}B\overline{C}$
   (d) ninguna de las respuestas anteriores
5. La expresión de salida para un circuito AND-OR-Inversor que consta de una puerta AND con entradas *A, B, C* y *D*, y otra puerta AND con las entradas *E* y *F* es
   (a) $ABCD + EF$
   (b) $\overline{A} + \overline{B} + \overline{C} + \overline{D} + \overline{E} + \overline{F}$
   (c) $\overline{(A + B + C + D)(E + F)}$
   (d) $(\overline{A} + \overline{B} + \overline{C} + \overline{D})(\overline{E} + \overline{F})$
6. Una función OR-Exclusiva se expresa como:
   (a) $\overline{A}\overline{B} + AB$
   (b) $\overline{A}B + A\overline{B}$
   (c) $(\overline{A} + B)(A + \overline{B})$
   (d) $(\overline{A} + \overline{B}) + (A + B)$
7. La operación AND se puede generar con
   (a) dos puertas NAND
   (b) tres puertas NAND
   (c) una puerta NOR
   (d) tres puertas NOR
8. La operación OR se puede generar con
   (a) dos puertas NOR
   (b) tres puertas NAND
   (c) cuatro puertas NAND
   (d) (a) y (b)
9. Cuando se usan símbolos duales en un diagrama lógico
   (a) las salidas invertidas se conectan con las entradas invertidas
   (b) los símbolos NAND generan operaciones AND
   (c) los símbolos negativa-OR generan operaciones OR
   (d) todas las respuestas son verdaderas
   (e) ninguna respuesta es verdadera
10. Las expresiones booleanas se pueden implementar con:
    (a) sólo puertas NAND
    (b) sólo puertas NOR

(c) combinaciones de puertas NAND y NOR
(d) combinaciones de puertas AND, puertas OR e inversores
(e) todas la anteriores

■ PROBLEMAS SECCIÓN 5.1 Circuitos lógicos combinacionales especiales

1. Dibujar el diagrama lógico con símbolos distintivos de un circuito AND-OR-Inversor de 4 entradas y triple anchura. Dibujar también el diagrama utilizando los símbolos rectangulares.

2. Escribir la expresión de salida de los circuitos de la Figura 5.47.

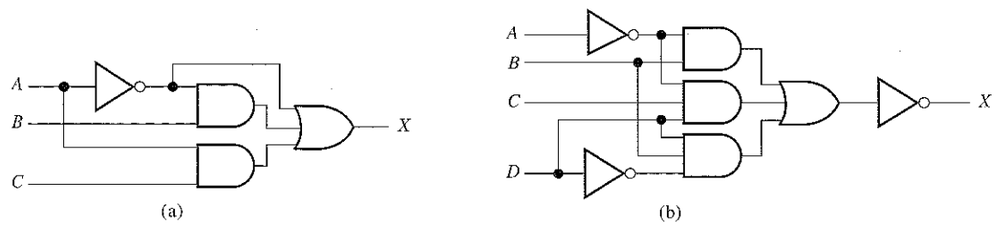


FIGURA 5.47

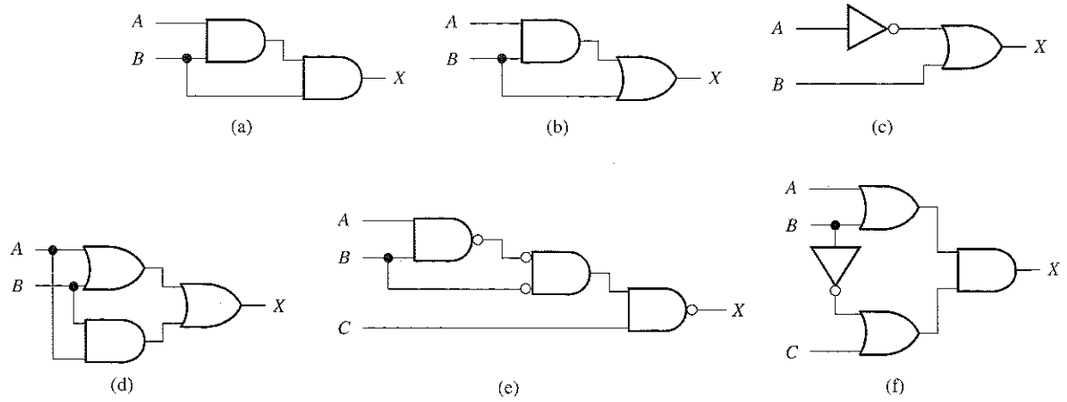


FIGURA 5.48

3. Escribir la expresión de salida de los circuitos de la Figura 5.48.

4. Escribir la expresión de salida de los circuitos de la Figura 5.49, y dibujar los circuitos equivalentes utilizando la configuración AND-OR.

5. Desarrollar la tabla de verdad de cada uno de los circuitos de la Figura 5.48.

6. Desarrollar la tabla de verdad de cada uno de los circuitos de la Figura 5.49.

7. Demostrar que un circuito NOR-exclusiva genera una salida que es un producto de sumas.

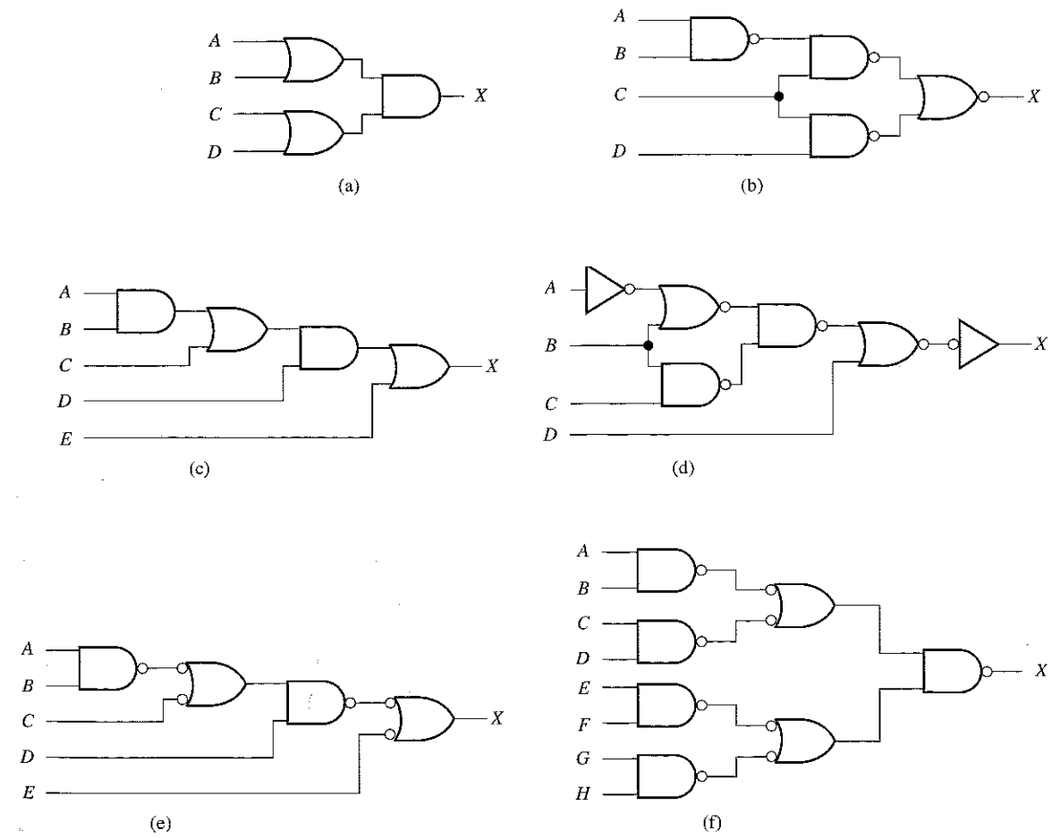


FIGURA 5.49

SECCIÓN 5.2 Implementación de la lógica combinacional

8. Utilizar puertas AND, puertas OR o combinaciones de ambas para implementar las siguientes expresiones lógicas:

(a) $X = AB$
(b) $X = A + B$
(c) $X = AB + C$
(d) $X = ABC + D$
(e) $X = A + B + C$
(f) $X = ABCD$
(g) $X = A(CD + B)$
(h) $X = AB(C + DEF) + CE(A + B + F)$

9. Utilizar puertas AND, puertas OR e inversores para implementar las siguientes expresiones lógicas:

(a) $X = AB + \overline{B}C$
(b) $X = A(B + \overline{C})$
(c) $X = A\overline{B} + AB$
(d) $X = \overline{ABC} + B(EF + \overline{G})$
(e) $X = A[BC(A + B + C + D)]$
(f) $X = B(C\overline{D}E + \overline{E}FG)(\overline{AB} + C)$

10. Utilizar puertas NAND, puertas NOR o combinaciones de ambas para implementar las siguientes expresiones lógicas:

(a) $X = \overline{A}B + CD + (\overline{A + B})(ACD + \overline{BE})$
(b) $X = AB\overline{C}\,\overline{D} + D\overline{E}F + \overline{A}\overline{F}$
(c) $X = \overline{A}[B + \overline{C}(D + E)]$

11. Implementar un circuito lógico para la tabla de verdad de la Tabla 5.7.

TABLA 5.7

Entradas			Salida
A	*B*	*C*	*X*
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

12. Implementar un circuito lógico para la tabla de verdad de la Tabla 5.8.

TABLA 5.8

Entradas				Salida
A	*B*	*C*	*D*	*X*
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

13. Simplificar el circuito de la Figura 5.50 tanto como sea posible, y verificar que el circuito simplificado es equivalente al original, demostrando que las tablas de verdad son idénticas.

14. Repetir el problema 13 para el circuito de la Figura 5.51.

15. Minimizar las puertas requeridas para implementar las funciones de cada apartado del problema 9 en forma de suma de productos.

16. Minimizar las puertas requeridas para implementar las funciones de cada apartado del problema 10 en forma de suma de productos.

17. Minimizar las puertas requeridas para implementar la función de los circuitos de cada apartado de la Figura 5.49 en forma de suma de productos.

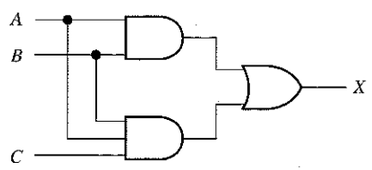


FIGURA 5.50

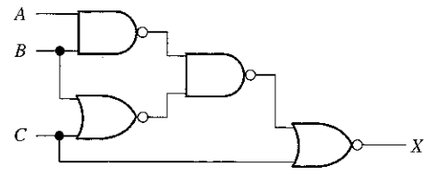


FIGURA 5.51

SECCIÓN 5.3 La propiedad universal de las puertas NAND y NOR

18. Implementar los circuitos lógicos de la Figura 5.47 utilizando sólo puertas NAND.
19. Implementar los circuitos lógicos de la Figura 5.51 utilizando sólo puertas NAND.
20. Repetir el problema 18 utilizando sólo puertas NOR.
21. Repetir el problema 19 utilizando sólo puertas NOR.

SECCIÓN 5.4 Lógica combinacional utilizando puertas NAND y NOR

22. Demostrar que las siguientes expresiones pueden implementarse utilizando puertas NOR:

(a) $X = ABC$
(b) $X = \overline{ABC}$
(c) $X = A + B$
(d) $X = A + B + \overline{C}$
(e) $X = \overline{AB} + \overline{CD}$
(f) $X = (A + B)(C + D)$
(g) $X = AB[C(\overline{DE} + \overline{AB}) + \overline{BCE}]$

23. Repetir el problema 22 utilizando sólo puertas NAND.
24. Implementar las funciones del problema 8 utilizando sólo puertas NAND.
25. Implementar las funciones del problema 9 utilizando sólo puertas NAND.

SECCIÓN 5.5 Funcionamiento con trenes de impulsos

26. Dados el circuito lógico y las formas de onda de entrada de la Figura 5.52, dibujar la forma de onda de salida.
27. Para el circuito lógico de la Figura 5.53, dibujar la forma de onda de salida con respecto a las entradas.

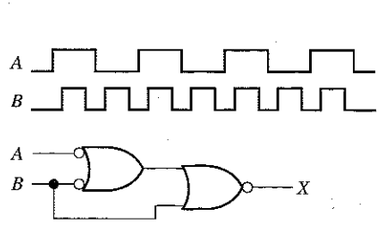


FIGURA 5.52

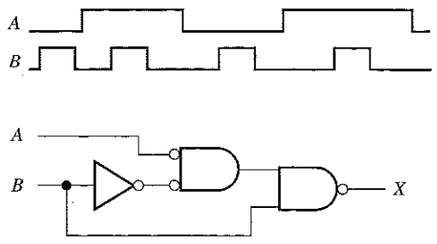


FIGURA 5.53

28. Para las formas de onda de entrada de la Figura 5.54, ¿qué circuito lógico generará la señal de salida mostrada?

29. Repetir el problema 28 para la señal de la Figura 5.55.

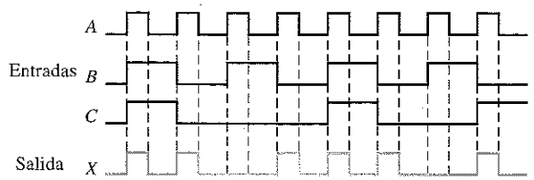


FIGURA 5.54

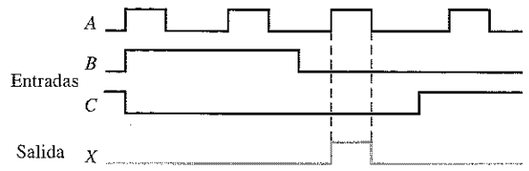


FIGURA 5.55

30. Para el circuito de la Figura 5.56, dibujar las formas de onda para los puntos numerados, indicando la relación de tiempos entre ellos.

31. Suponiendo un tiempo de propagación en cada puerta de 10 nanosegundos (ns), determinar si las entradas indicadas generarán la forma de onda de salida *X deseada* de la Figura 5.57 (impulso con un mínimo $t_w = 25$ ns como el mostrado).

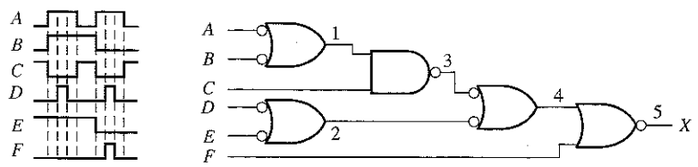


FIGURA 5.56

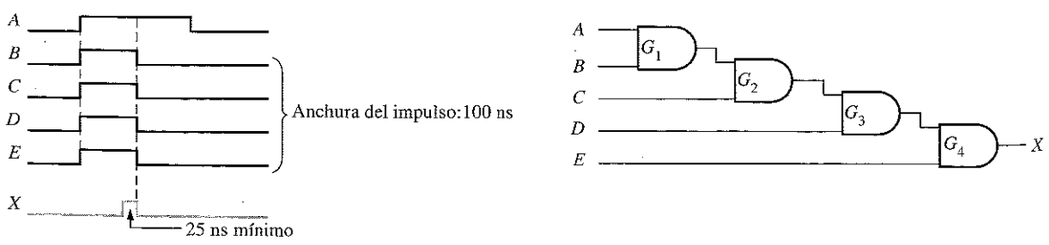


FIGURA 5.57

SECCIÓN 5.6 Localización de averías

32. Para el circuito lógico y la señal de entrada de la Figura 5.58, se observa la señal de salida indicada. Determinar si esta señal de salida es correcta.

33. La forma de onda de salida de la Figura 5.59 es incorrecta para las entradas que se aplican al circuito. Suponiendo que una puerta del circuito está fallando, con su salida a un nivel ALTO o BAJO constante, determinar la puerta que falla y el tipo de fallo (circuito abierto o cortocircuito).

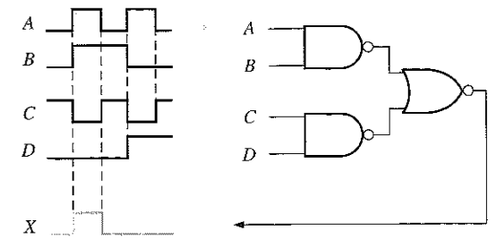


FIGURA 5.58

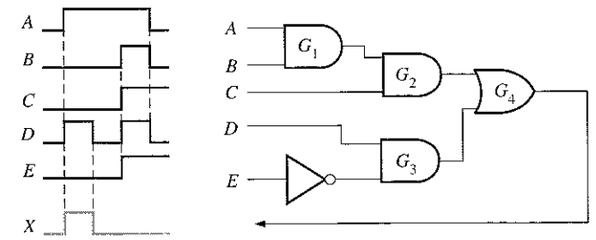


FIGURA 5.59

34. Repetir el problema 33 para el circuito de la Figura 5.60 para las señales de entrada y salida dadas.

35. El nodo de la Figura 5.61 está a nivel BAJO. Un pulsador lógico y un indicador de corriente se aplican en los puntos que se indican en el diagrama. A partir de estas observaciones, ¿qué se puede concluir?

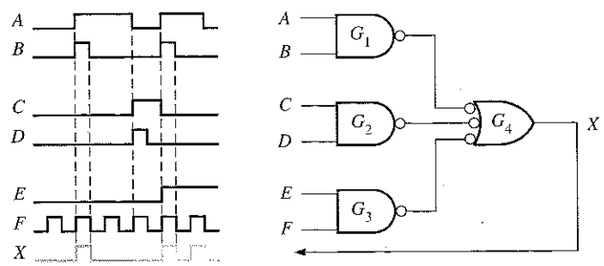
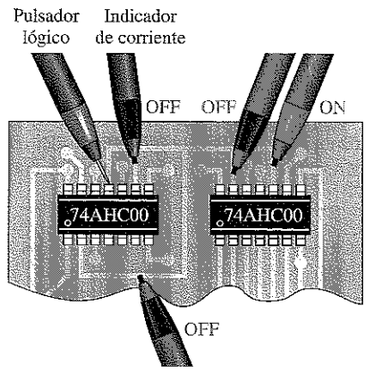
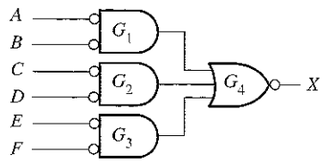


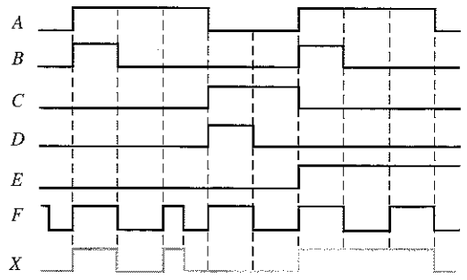
FIGURA 5.60

FIGURA 5.61





(a)



(b)

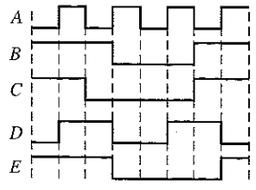
FIGURA 5.62

36. La Figura 5.62(a) es una red lógica bajo prueba. La Figura 5.62(b) muestra las formas de onda que se observan en el analizador lógico. Para las entradas que se aplican al circuito, la salida es incorrecta. Suponiendo que una puerta de la red ha fallado, estando su entrada a nivel alto o a nivel bajo constante, determinar la puerta que falla y el tipo de fallo.

37. Al circuito lógico de la Figura 5.63 se le aplican las formas de onda de entrada mostradas.

(a) Determinar la señal de salida correcta con respecto a las entradas.
(b) Determinar la señal de salida si la salida de la puerta G_3 está en circuito abierto.
(c) Determinar la señal de salida si la entrada superior de la puerta G_3 está cortocircuitada a masa.

38. El circuito lógico de la Figura 5.64 tiene disponible un único punto de prueba intermedio próximo a la salida. Para las entradas indicadas, se observa la señal dada en el punto de prueba. ¿Es correcta esta forma de onda? Si no lo es, ¿cuáles son los posibles fallos que podrían generar dicha señal?



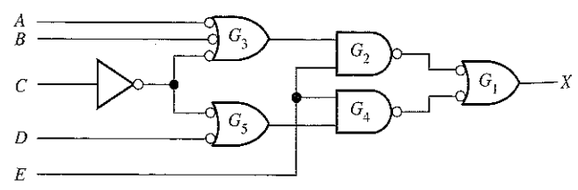
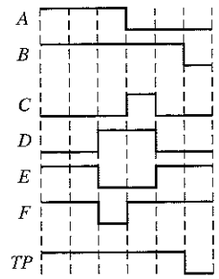


FIGURA 5.63



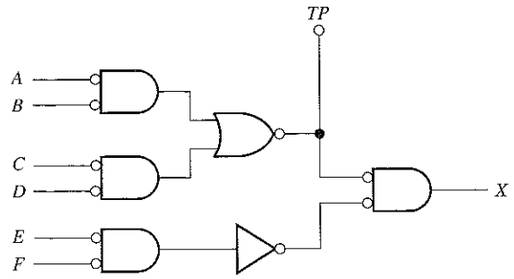


FIGURA 5.64

SECCIÓN 5.7 Aplicación a los sistemas digitales

39. Implementar la lógica del motor de lubricación de la Figura 5.45 con puertas NOR e inversores.

40. Repetir las actividades 1 a 3 del banco de trabajo digital 1 utilizando puertas NOR e inversores.

41. Modificar la lógica de control de los motores para que, en la condición $S_1\ S_2\ S_3\ S_4 = 0101$, se detenga la cinta transportadora pero la sierra de corte continúe funcionando.

Problemas especiales de diseño

42. Diseñar un circuito lógico para generar una salida a nivel ALTO si y sólo si la entrada, representada por un número binario de 4 bits, es mayor que doce o menor que tres. Desarrollar primero la tabla de verdad y después dibujar el diagrama lógico.

43. Desarrollar el circuito lógico que cumpla los siguientes requerimientos:

Una lámpara situada en una habitación puede accionarse mediante dos interruptores, uno colocado detrás de la puerta y el otro frente a la puerta. La lámpara se enciende si se activa el interruptor frente a la puerta y el de detrás de la misma no se activa, o en el caso contrario. La lámpara está apagada si ambos interruptores están desactivados o si ambos están activados. Una salida a nivel ALTO representa una condición de encendido y una salida a nivel BAJO representa la condición de apagado.

44. Diseñar un circuito que se añadirá a la lógica de control desarrollada en el banco de trabajo digital 1, y que indique una condición de desactivación mediante una luz roja.

45. Desarrollar el diagrama lógico NAND para un codificador hexadecimal de teclado que convierta cada pulsación a binario.

Localización de averías en el banco de trabajo electrónico

46. Abrir el archivo PR05_46.EWB y probar el circuito lógico para determinar si existe un fallo. Si existe, identificarlo cuando sea posible.

47. Abrir el archivo PR05_47.EWB y probar el circuito lógico para determinar si existe un fallo. Si existe, identificarlo cuando sea posible.

48. Abrir el archivo PR05_48.EWB y probar el circuito lógico para determinar si existe un fallo. Si existe, identificarlo cuando sea posible.

49. Abrir el archivo PR05_49.EWB y probar el circuito lógico para determinar si existe un fallo. Si existe, identificarlo cuando sea posible.

■ RESPUESTAS A LAS REVISIONES DE CADA SECCIÓN

SECCIÓN 5.1

1. **(a)** $\overline{A\overline{B} + \overline{C}D} = \overline{1 \cdot \overline{0} + \overline{1} \cdot 0} = 1$ **(b)** $\overline{A\overline{B} + \overline{C}D} = \overline{1 \cdot \overline{1} + \overline{0} \cdot \overline{1}} = 0$
   (c) $\overline{A\overline{B} + \overline{C}D} = \overline{0 \cdot \overline{1} + \overline{1} \cdot \overline{1}} = 0$
2. **(a)** $A\overline{B} + \overline{A}B = 1 \cdot \overline{0} + \overline{1} \cdot 0 = 1$ **(b)** $A\overline{B} + \overline{A}B = 1 \cdot \overline{1} + \overline{1} \cdot 1 = 0$
   (c) $A\overline{B} + \overline{A}B = 0 \cdot \overline{1} + \overline{0} \cdot 1 = 1$ **(d)** $A\overline{B} + \overline{A}B = 0 \cdot \overline{0} + \overline{0} \cdot 0 = 0$
3. $X = 1$ cuando $ABC = 000, 011, 101, 110$ y 111; $X = 0$ cuando $ABC = 001, 010$ y 100.
4. $X = AB + \overline{A}\,\overline{B}$; el circuito está constituido por dos puertas AND, una puerta OR y dos inversores. Consultar el diagrama de la Figura 5.8(b).

SECCIÓN 5.2

1. **(a)** $X = ABC + AB + AC$: tres puertas AND y una puerta OR.
   (b) $X = AB(C + DE)$: tres puertas AND y una puerta OR.
2. $X = ABC + \overline{A}\,\overline{B}\,\overline{C}$: dos puertas AND, una puerta OR y tres inversores.
3. **(a)** $X = AB(C + 1) + AC = AB + AC$.
   (b) $X = AB\,(C + DE) = ABC + ABDE$.

SECCIÓN 5.3

1. **(a)** $X = \overline{A} + B$: es una puerta NAND con A y $\overline{B}$ en sus entradas.
   (b) $X = A\,\overline{B}$: es una puerta NAND con A y $\overline{B}$ en sus entradas, seguida de una puerta NAND utilizada como inversor.
2. **(a)** $X = \overline{A} + B$: es una puerta NOR con $\overline{A}$ y B en sus entradas, seguida de una puerta NOR utilizada como inversor.
   (b) $X = A\overline{B}$: es una puerta NOR con $\overline{A}$ y B en sus entradas.

SECCIÓN 5.4

1. $X = \overline{(\overline{A} + \overline{B} + \overline{C})DE}$: es una puerta NAND con las entradas A, B y C, con su salida conectada a una segunda puerta NAND de 3 entradas junto con las entradas D y E.
2. $X = \overline{\overline{A}\,\overline{B}\,\overline{C} + (D + E)}$: es una puerta NOR con las entradas A, B y C, con su salida conectada a una segunda puerta NOR de 3 entradas junto con las entradas D y E.

SECCIÓN 5.5

1. La salida de la puerta OR-exclusiva es un impulso de 15 μs seguido de un impulso de 25 μs, con una separación de 10 μs entre los impulsos.
2. La salida de la puerta NOR-exclusiva es un nivel alto cuando ambas entradas están a nivel alto, o cuando ambas entradas están a nivel bajo.

SECCIÓN 5.6

1. Los fallos más comunes en las puertas son entrada o salida en circuito abierto, y entrada o salida cortocircuitada a masa.
2. La entrada cortocircuitada a V_{CC} hace que la salida se mantenga a nivel bajo.
3. **(a)** La salida de G_4 está a nivel alto hasta el flanco de bajada del séptimo impulso; luego pasa a nivel bajo.
   (b) La salida de G_4 es igual a la entrada D.
   (c) La salida de G_4 es la misma que la salida de G_2, mostrada en la Figura 5.43(b).

SECCIÓN 5.7

1. La lógica de control decodifica las entradas de los interruptores y genera las salidas de control para el motor.

2. Se trata de una condición permitida. La sierra de cinta funciona.
3. Se trata de una condición no permitida, ya que la cinta transportadora y la sierra de corte no se pueden activar al mismo tiempo, y también porque la cinta transportadora está funcionando sin lubricación.

■ RESPUESTAS A LOS PROBLEMAS RELACIONADOS DE LOS EJEMPLOS

5.1 $Y = AB + AC + BC$

5.2 $Y = \overline{AB + AC + BC}$

Si $A = 0$ y $B = 0$, $Y = \overline{0 \cdot 0 + 0 \cdot 1 + 0 \cdot 1} = \overline{0} = 1$

Si $A = 0$ y $C = 0$, $Y = \overline{0 \cdot 1 + 0 \cdot 0 + 1 \cdot 0} = \overline{0} = 1$

Si $B = 0$ y $C = 0$, $Y = \overline{1 \cdot 0 + 1 \cdot 0 + 0 \cdot 0} = \overline{0} = 1$

5.3 No se puede simplificar.

5.4 No se puede simplificar.

5.5 $X = A + B + C + D$ es válido.

5.6 Véase la Figura 5.65.

5.7 $X = \overline{(\overline{AB}C)(\overline{DE}F)} = (\overline{AB})C + (\overline{DE})F = (\overline{A} + \overline{B})C + (\overline{D} + \overline{E})F$

5.8 Véase la Figura 5.66.

5.9 $X = \overline{(\overline{A + B} + C) + (\overline{D + E} + F)} = (\overline{A + B} + C)(\overline{D + E} + F) = (\overline{A}\,\overline{B} + C)(\overline{D}\,\overline{E} + F)$

5.10 Véase la Figura 5.67.

5.11 Véase la Figura 5.68.

5.12 Véase la Figura 5.69.

5.13 Véase la Figura 5.70.

5.14 Véase la Figura 5.71.

FIGURA 5.65

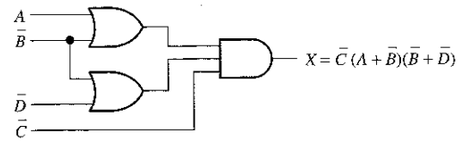


FIGURA 5.66

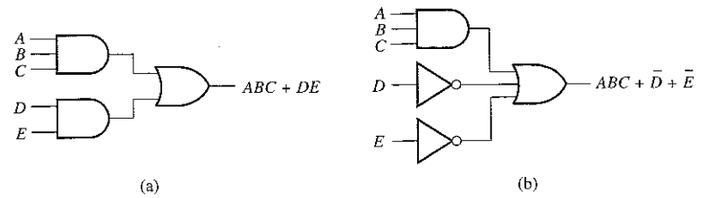


FIGURA 5.67

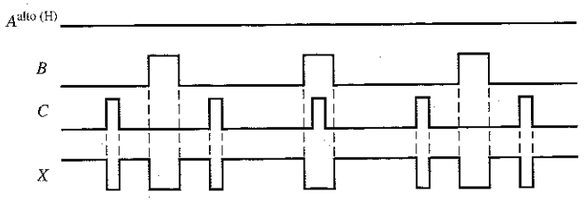


FIGURA 5.68

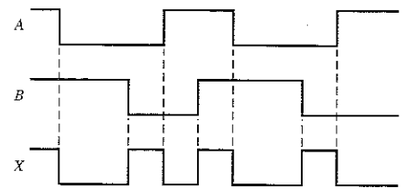


FIGURA 5.69

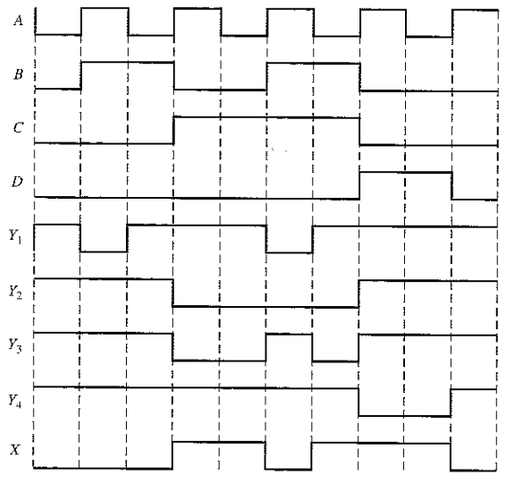


FIGURA 5.70

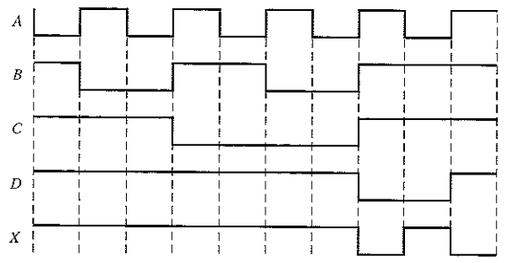
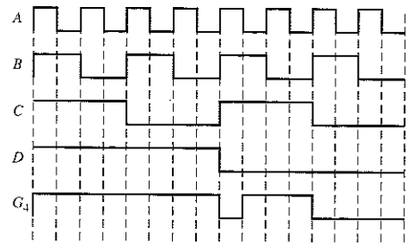


FIGURA 5.71



RESPUESTAS AL AUTOTEST

1. (d) **2.** (b) **3.** (c) **4.** (a) **5.** (d) **6.** (b) **7.** (a) **8.** (d) **9.** (d) **10.** (e)

6

FUNCIONES DE LA LÓGICA COMBINACIONAL

6.1 Sumadores básicos
6.2 Sumadores binarios en paralelo
6.3 Comparadores
6.4 Decodificadores
6.5 Codificadores
6.6 Convertidores de código
6.7 Multiplexores (selectores de datos)
6.8 Demultiplexores
6.9 Generadores/comprobadores de paridad
6.10 Localización de averías
6.11 Aplicación a los sistemas digitales

Simulación por computadora de circuitos digitales:
El tutorial de Electronics Workbench (EWB) se encuentra en la dirección http://www.prenhall.com/floyd.

■ OBJETIVOS DEL CAPÍTULO

- ❑ Distinguir entre semisumadores y sumadores completos.
- ❑ Utilizar sumadores completos para implementar sumadores en paralelo binarios de múltiples bits.
- ❑ Explicar las diferencias entre sumadores con acarreo serie y sumadores paralelo con acarreo anticipado.
- ❑ Utilizar los comparadores de magnitud para determinar la relación entre dos números binarios y utilizar los comparadores en cascada para realizar comparaciones de números más grandes.
- ❑ Implementar un decodificador binario básico.
- ❑ Utilizar decodificadores BCD a 7 segmentos en sistemas con displays.
- ❑ Emplear un codificador de prioridad BCD-binario en un sencillo sistema con teclado.
- ❑ Convertir, utilizando dispositivos lógicos, números en código BCD a números binarios, números binarios a código Gray, y códigos Gray a números binarios.
- ❑ Aplicar multiplexores para la selección de datos, los displays multiplexados, la generación de funciones lógicas y sistemas sencillos de comunicaciones.
- ❑ Utilizar decodificadores como demultiplexores.
- ❑ Explicar el significado de paridad.
- ❑ Usar generadores y comprobadores de paridad para detectar errores de bits en los sistemas digitales.
- ❑ Implementar un sencillo sistema de comunicación de datos.
- ❑ Identificar glitches (impulsos no deseados de muy corta duración), errores muy comunes en los sistemas digitales.

■ INTRODUCCIÓN

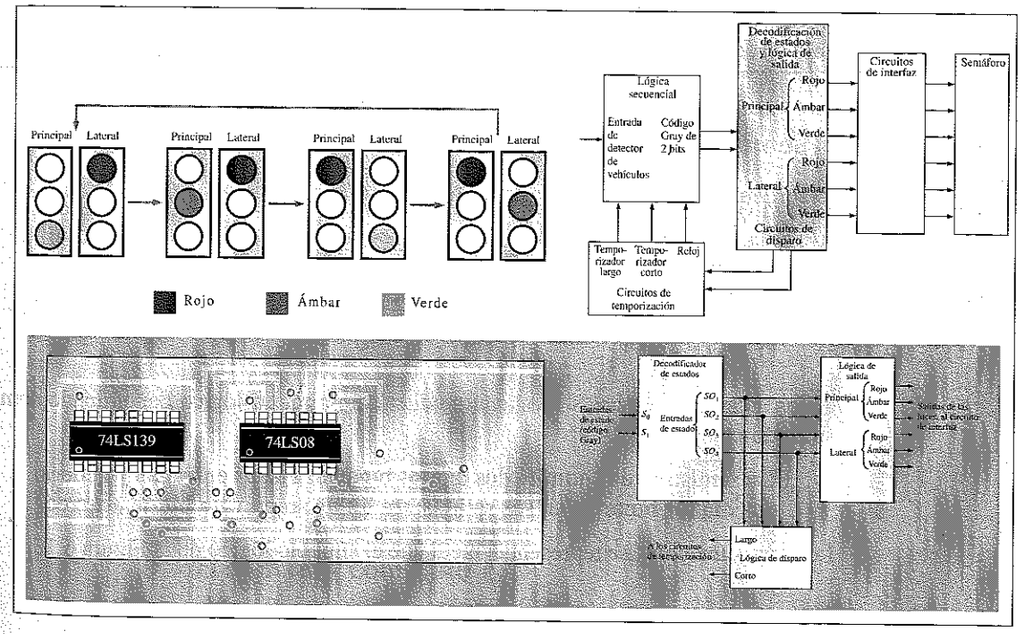
En este capítulo se presentan distintos tipos de circuitos lógicos combinacionales MSI (Medium Scale Integration, integración a media escala), incluyendo sumadores, comparadores, decodificadores, codificadores, convertidores de código, multiplexores (selectores de datos), demultiplexores y generadores/comprobadores de paridad. También se incluyen ejemplos de aplicación de muchos de estos dispositivos para demostrar cómo pueden usarse en situaciones prácticas.

DISPOSITIVOS ESPECÍFICOS

74HC42	74LS47	74LS83A
74HC85	74HC138	74LS139
74HC147	74F148	74HC151
74HC154	74HC157A	74184
74185	74LS280	74LS283

■ APLICACIÓN A LOS SISTEMAS DIGITALES

Esta aplicación a un sistema digital ilustra los conceptos que se enseñan en este capítulo y consiste en una parte de un sistema de control de semáforos. La aplicación del siguiente capítulo presenta otro método para este mismo sistema, mientras que en los Capítulos 8, 9 y 11 nos centraremos en otras partes del mismo. Básicamente, este sistema se encarga de controlar el tráfico en la intersección de una calle con mucho tráfico y otra más despejada. El sistema está formado por una parte en la que se aplica la lógica combinacional, explicada en este capítulo y en el Capítulo 5, un circuito de temporización, que se verá en el Capítulo 8 y un sistema de lógica secuencial, al que se aplican los conceptos de los Capítulos 8 y 9.



6.1 ■ SUMADORES BÁSICOS

Los sumadores son muy importantes no solamente en las computadoras, sino en muchos tipos de sistemas digitales en los que se procesan datos numéricos. Comprender el funcionamiento de un sumador es fundamental en el estudio de los sistemas digitales. En esta sección estudiaremos el semisumador y el sumador completo.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir el funcionamiento de un semisumador. ■ Dibujar el diagrama lógico de un semisumador. ■ Describir el funcionamiento de un sumador completo. ■ Dibujar el diagrama lógico de un sumador completo utilizando semisumadores. ■ Implementar un sumador completo mediante lógica AND-OR.

El semisumador

Recordemos las reglas básicas de la suma binaria expuestas en el Capítulo 2:

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10$$

Todas estas operaciones se realizan mediante un circuito lógico denominado **semisumador**.

Un semisumador admite dos dígitos binarios en sus entradas y genera dos dígitos binarios en sus salidas: un bit de suma y un bit de acarreo.

Los semisumadores se representan mediante el símbolo lógico de la Figura 6.1.

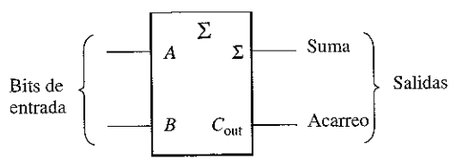


FIGURA 6.1
Símbolo lógico de un semisumador.

Lógica del semisumador. A partir del funcionamiento lógico de un semisumador, expuesto en la Tabla 6.1, las expresiones correspondientes a la suma y al acarreo de salida se pueden obtener como funciones de las entradas. Obsérvese que la salida de acarreo (C_{out}) es 1 sólo cuando A y B son 1; por tanto, C_{out} puede expresarse como una operación AND de las variables de entrada.

$$C_{out} = AB \tag{6.1}$$

Obsérvese ahora que la salida correspondiente a la suma (Σ) es 1 sólo si las variables A y B son distintas. Por tanto, la suma puede expresarse como una operación OR-exclusiva de las variables de entrada.

$$\Sigma = A \oplus B \tag{6.2}$$

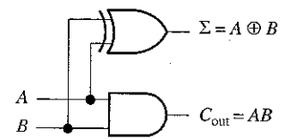
TABLA 6.1
Tabla de verdad de un semisumador.

A	B	C_{out}	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Σ = suma
C_{out} = acarreo de salida
A y B = variables de entrada (operandos)

A partir de las ecuaciones (6.1) y (6.2), se puede desarrollar la implementación lógica del funcionamiento de un semisumador. La salida de acarreo se produce mediante una puerta AND, siendo A y B sus dos entradas, y la salida de la suma se obtiene mediante una puerta OR-exclusiva, como muestra la Figura 6.2.

FIGURA 6.2
Diagrama lógico de un semisumador.



El sumador completo

El segundo tipo de sumador es el **sumador completo**.

Un sumador acepta dos bits de entrada y un acarreo de entrada, y genera una salida de suma y un acarreo de salida.

La diferencia principal entre un sumador completo y un semisumador es que el sumador completo acepta un acarreo de entrada. El símbolo lógico de un sumador completo se muestra en la Figura 6.3, y la tabla de verdad mostrada en la Tabla 6.2 describe su funcionamiento.

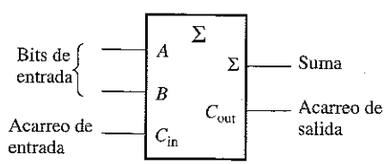


FIGURA 6.3
Símbolo lógico de un sumador completo.

TABLA 6.2
Tabla de verdad de un sumador completo.

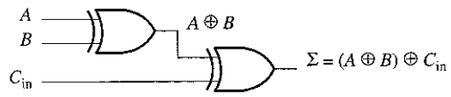
A	B	C_{in}	C_{out}	Σ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

C_{in} = acarreo de entrada, algunas veces se designa por CI
C_{out} = acarreo de salida, algunas veces se designa por CO
Σ = suma
A y B = variables de entrada (operandos)

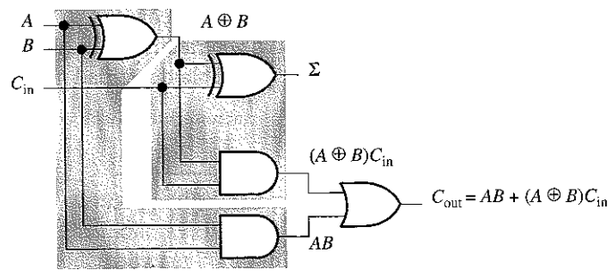
Lógica del sumador completo. Un sumador completo suma los dos bits de entrada y el bit de acarreo de entrada. A partir del semisumador, ya conocemos que la suma de los dos bits de entrada A y B consiste en la operación OR-exclusiva entre estas dos variables, $A \oplus B$. Para sumar el acarreo de entrada (C_{in}) a los bits de entrada, hay que volver a aplicar la operación OR-exclusiva, obteniéndose la siguiente ecuación de salida para el sumador completo:

$$\Sigma = (A \oplus B) \oplus C_{in} \tag{6.3}$$

Esto significa que, para implementar la función de un sumador completo, se pueden utilizar dos puertas OR-exclusiva. La primera tiene que generar el término $A \oplus B$, y la segunda toma como entradas la salida de la primera puerta XOR y el acarreo de entrada, como se muestra en la Figura 6.4(a).



(a) Lógica necesaria para realizar la suma de tres bits



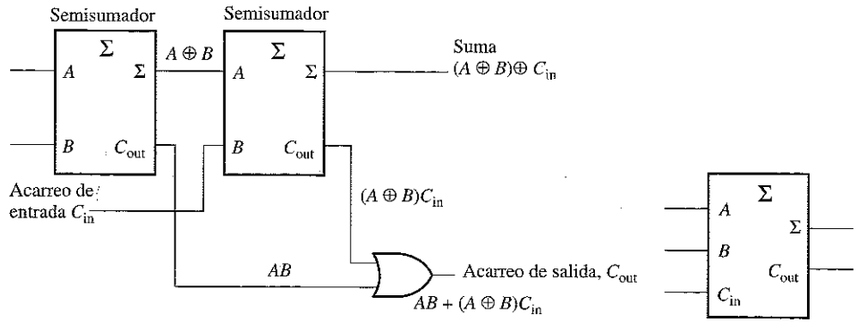
(b) Circuito lógico de un sumador completo (cada semisumador se representa por un área sombreada)

FIGURA 6.4
Lógica de un sumador completo.

El acarreo de salida es 1 cuando las dos entradas de la primera puerta XOR son 1, o cuando las dos entradas de la segunda puerta XOR son 1. Esto se puede comprobar analizando la Tabla 6.2. El acarreo de salida del sumador completo se obtiene a partir del producto lógico (AND) de las entradas A y B, y del producto lógico (AND) de $A \oplus B$ y de C_{in}, sumando (OR) después ambos términos resultantes, como se muestra en la ecuación (6.4). Esta función, una vez implementada, se combina con la de la suma lógica para constituir un circuito sumador completo, como se muestra en la Figura 6.4(b).

$$C_{out} = AB + (A \oplus B)\,C_{in} \tag{6.4}$$

Obsérvese que, en la Figura 6.4(b), existen dos semisumadores conectados, como se muestra en el diagrama de bloques de la Figura 6.5(a), cuyos acarreos de salida se aplican a una puerta OR. El símbolo lógico mostrado en la Figura 6.5(b) será el que normalmente empleemos para representar un sumador completo.

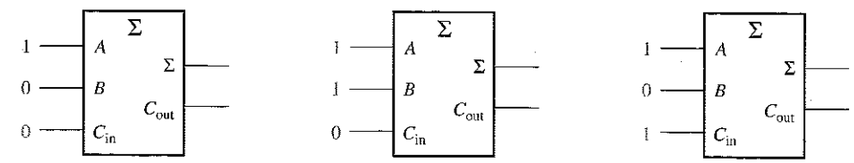


(a) Dos semisumadores formando un sumador completo

(b) Símbolo lógico de un sumador completo

FIGURA 6.5
Sumador completo implementado mediante semisumadores.

EJEMPLO 6.1 Para cada uno de los tres sumadores completos de la Figura 6.6, determinar las salidas para las entradas indicadas.



(a) (b) (c)

FIGURA 6.6

Solución

(a) Los bits de entrada son $A = 1$, $B = 0$ y $C_{in} = 0$.

$$1 + 0 + 0 = 1 \text{ sin acarreo}$$

Por tanto, $\Sigma = 1$ y $C_{out} = 0$

(b) Los bits de entrada son $A = 1$, $B = 1$ y $C_{in} = 0$.

$$1 + 1 + 0 = 0 \text{ con acarreo de } 1$$

Por tanto, $\Sigma = 0$ y $C_{out} = 1$

(c) Los bits de entrada son $A = 1$, $B = 0$ y $C_{in} = 1$.

$$1 + 0 + 1 = 0 \text{ con acarreo de } 1$$

Por tanto, $\Sigma = 0$ y $C_{out} = 1$

Problema relacionado ¿Cuáles serán las salidas del sumador completo para $A = 1$, $B = 1$ y $C_{in} = 1$?

REVISIÓN DE LA SECCIÓN 6.1

1. Determinar la suma (Σ) y el acarreo de salida (C_{out}) de un semisumador para cada uno de los siguientes grupos de bits de entrada:
   (a) 01 **(b)** 00 **(c)** 10 **(d)** 11
2. Un sumador completo tiene $C_{in} = 1$. ¿Cuánto vale la suma (Σ) y el acarreo de salida (C_{out}) cuando $A = 1$ y $B = 1$?

6.2 ■ SUMADORES BINARIOS EN PARALELO

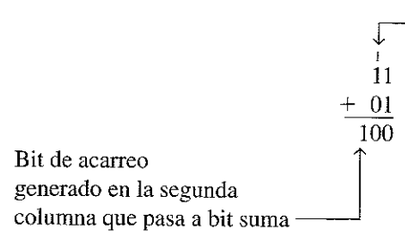
Para formar un sumador binario en paralelo se conectan dos o más sumadores completos. En esta sección aprenderemos los principios básicos de este tipo de sumador, de manera que podamos entender todas las funciones necesarias de entrada y salida cuando se trabaja con este tipo de dispositivos.

Al finalizar esta sección, el lector deberá ser capaz de:

- Utilizar sumadores completos para implementar un sumador binario en paralelo.
- Explicar el proceso de adición en un sumador binario en paralelo. ■ Emplear la tabla de verdad para un sumador en paralelo de 4 bits ■ Utilizar los dispositivos 74LS83A y 74LS283 para sumar dos números binarios de 4 bits. ■ Ampliar el sumador de 4 bits para poder realizar adiciones de 8 bits o 16 bits.

Como se ha visto en la Sección 6.1, un único sumador completo es capaz de sumar dos números binarios de 1 bit y un acarreo de entrada. Para sumar números binarios de más de un bit, se tienen que utilizar sumadores completos adicionales. Cuando se suman dos números binarios, cada columna genera un bit de suma y un 1 o 0, correspondiente al bit de acarreo, que se añade a la columna inmediata de la izquierda, como se muestra a continuación con dos números de 2 bits.

NOTAS INFORMÁTICAS
Las computadoras realizan la operación de suma con dos números a un tiempo, denominados *operandos*. El *operando fuente* es un número que se añade a un número existente denominado *operando de destino*, que es el que se almacena en un registro de la ALU, tal como el acumulador. A continuación, la suma de los dos números se almacena de nuevo en el acumulador. La adición se realiza con números enteros o números en coma flotante utilizando, respectivamente, las instrucciones ADD o FADD.



Para sumar dos números binarios, se necesita un sumador completo por cada bit que tengan los números que se quieren sumar. Así, para números de dos bits se necesitan dos sumadores; para números de cuatro bits hacen falta cuatro sumadores, y así sucesivamente. La salida de acarreo de cada sumador se conecta a la entrada de acarreo del sumador de orden inmediatamente superior, como se muestra en la Figura 6.7 para un sumador de 2 bits. Téngase en cuenta que se puede usar un semisumador para la posición menos significativa, o bien se puede poner a 0 (masa) la entrada de acarreo de un sumador completo, ya que no existe entrada de acarreo en la posición del bit menos significativo.

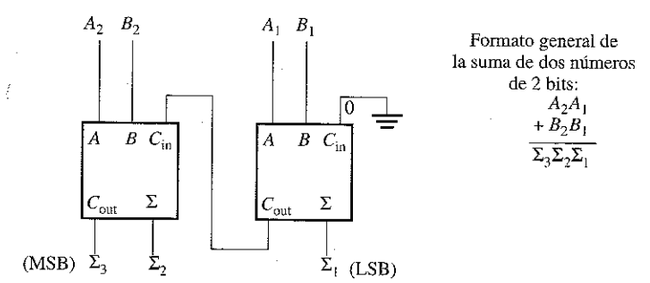
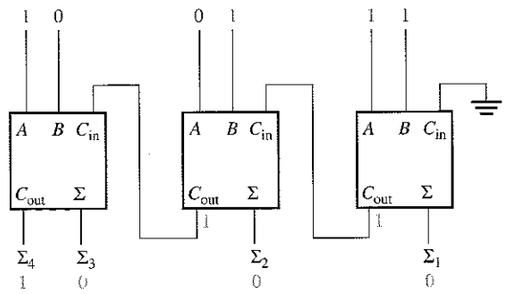


FIGURA 6.7
Diagrama de bloques de un sumador paralelo de 2 bits básico utilizando dos sumadores completos.

En la Figura 6.7 los bits menos significativos (LSB) de los dos números se representan como A_1 y B_1. Los siguientes bits de orden superior se representan como A_2 y B_2. Los tres bits de suma son Σ_1,Σ_2 y Σ_3. Obsérvese que el acarreo de salida del sumador completo de más a la izquierda se convierte en el bit más significativo (MSB) en la suma Σ_3.

EJEMPLO 6.2 Determinar la suma generada por el sumador paralelo de tres bits mostrado en la Figura 6.8 e indicar los acarreos intermedios cuando se están sumando los números 101 y 011.

FIGURA 6.8



Solución. Los bits menos significativos (LSB) de los dos números se suman en el sumador completo situado más a la derecha. En la Figura 6.8 se indican los bits de suma y los acarreos intermedios en tramado de gris.

Problema relacionado. Cuando se utiliza un sumador paralelo de 3 bits para sumar los números 111 y 101, ¿cuáles son las salidas de suma?

Sumadores en paralelo de cuatro bits

Un grupo de cuatro bits se denomina **nibble**. Un sumador básico en paralelo de 4 bits se implementa mediante cuatro sumadores completos, como se muestra en la Figura 6.9. De nuevo, los bits menos significativos (A_1 y B_1) de cada número que se suma, se introducen en el sumador completo que está más a la derecha; los bits de orden más alto se introducen sucesivamente en los siguientes sumadores, aplicando los bits más significativos de cada número (A_4 y B_4) al sumador que está más a la izquierda. La salida de acarreo de cada sumador se conecta a la entrada de acarreo del siguiente sumador de orden superior. Estos acarreos se denominan *acarreos internos*.

En la mayoría de las hojas de características suministradas por los fabricantes, se denomina C_0 al acarreo de entrada del sumador del bit menos significativo; C_4, en el caso de cuatro bits, sería el acarreo de salida del sumador del bit más significativo; Σ_1 (LSB) hasta Σ_4 (MSB) son las sumas de salida. El símbolo lógico correspondiente se muestra en la Figura 6.9(b).

En función del método utilizado para manipular los acarreos [1] en un sumador paralelo, existen dos tipos: el sumador de *acarreo serie* y el sumador *de acarreo anticipado*.

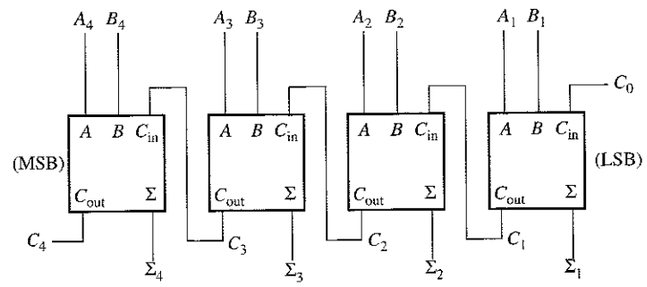
Un sumador **de acarreo serie** es aquel en el que la salida de acarreo de cada sumador completo se conecta a la entrada de acarreo de la siguiente etapa de orden inmediatamente superior (una etapa es un sumador completo). La suma y el acarreo de salida de cualquier etapa no se pueden generar hasta que tiene lugar el acarreo de entrada, lo que da lugar a un retardo temporal en el proceso de adición. El retardo de propagación del acarreo para cada sumador completo es el tiempo transcurrido desde la aplicación del acarreo de entrada hasta que se produce el acarreo de salida, suponiendo que las entradas A y B ya existan.

Un método que permite acelerar el proceso de adición eliminando este retardo del acarreo serie es la adición con **acarreo anticipado**. El sumador con acarreo anticipado anticipa el acarreo de salida de cada etapa y, en función de los bits de entrada

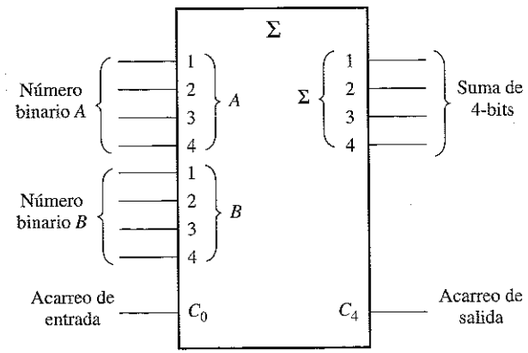
[1] Para más información véase Apéndice C.

de ambos sumandos, genera el acarreo de salida bien mediante la *generación de acarreo* o *la propagación de acarreo*.

FIGURA 6.9
Sumador en paralelo de 4 bits.



(a) Diagrama de bloques



(b) Símbolo lógico

La **generación de acarreo** tiene lugar cuando el sumador completo genera internamente un acarreo de salida. Sólo cuando ambos bits de entrada son 1 se genera un acarreo. El acarreo generado, C_g, se expresa como la función AND de los 2 bits de entrada, A y B.

$$C_g = AB$$

La **propagación de acarreo** tiene lugar cuando el acarreo de entrada se transmite como acarreo de salida. Un acarreo de entrada puede ser propagado por el sumador completo cuando uno o ambos bits de entrada son igual a 1. El acarreo propagado, C_p, se expresa como la función OR de los bits de entrada.

$$C_p = A + B$$

Tabla de verdad de un sumador en paralelo de 4 bits

La Tabla 6.3 es la tabla de verdad de un sumador de 4 bits. En algunas hojas de características, las tablas de verdad se denominan *tablas de función* o *tablas de verdad funcionales*. El subíndice n representa los bits del sumador y puede ser igual a 1, 2, 3 o 4

para un sumador de 4 bits. C_{n-1} es el acarreo del sumador previo. Los acarreos C_1, C_2 y C_3 se generan internamente. C_0 es un acarreo de entrada externo y C_4 es una salida. El Ejemplo 6.3 ilustra cómo utilizar la Tabla 6.3.

TABLE 6.3
Tabla de verdad para cada etapa de un sumador en paralelo de 4 bits.

C_{n-1}	A_n	B_n	Σ_n	C_n
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

EJEMPLO 6.3 Utilizar la tabla de verdad del sumador en paralelo de 4 bits (Tabla 6.3) para encontrar la suma y el acarreo de salida correspondientes a los siguientes dos números binarios de 4 bits, siendo el acarreo de entrada (C_{n-1}) igual a 0:

$$A_4A_3A_2A_1 = 1100 \text{ y } B_4B_3B_2B_1 = 1100$$

Solución. Para $n = 1$: $A_1 = 0$, $B_1 = 0$ y $C_{n-1} = 0$. Según la primera fila de la tabla,

$$\Sigma_1 = \mathbf{0} \text{ y } C_1 = 0$$

Para $n = 2$: $A_2 = 0$, $B_2 = 0$ y $C_{n-1} = 0$. Según la primera fila de la tabla,

$$\Sigma_2 = \mathbf{0} \text{ y } C_2 = 0$$

Para $n = 3$: $A_3 = 1$, $B_3 = 1$ y $C_{n-1} = 0$. Según la cuarta fila de la tabla,

$$\Sigma_3 = \mathbf{0} \text{ y } C_3 = 1$$

Para $n = 4$: $A_4 = 1$, $B_4 = 1$ y $C_{n-1} = 1$. Según la última fila de la tabla,

$$\Sigma_4 = \mathbf{1} \text{ y } C_4 = \mathbf{1}$$

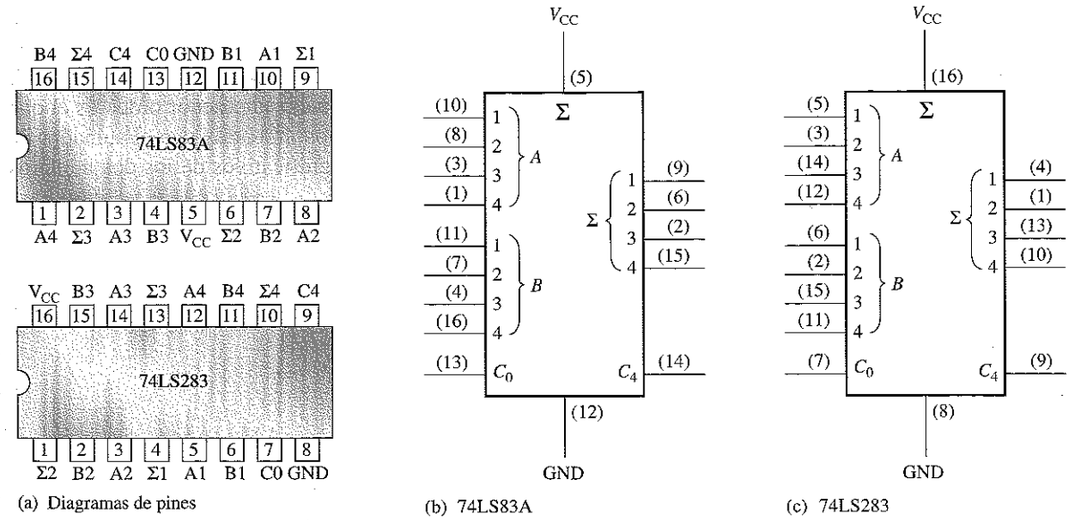
C_4 será el acarreo de salida; la suma de 1100 y 1100 es 11000.

Problema relacionado. Utilizar la tabla de verdad (Tabla 6.3) para calcular la suma de los números binarios 1011 y 1010.

Sumadores MSI

Ejemplos de sumadores paralelo de 4 bits que están disponibles como circuitos integrados de media escala (MSI) son los dispositivos TTL Schottky de baja potencia 74LS83A y 74LS283. Estos dos dispositivos son funcionalmente idénticos entre sí, aunque no son compatibles en cuanto a la disposición de sus pines; es decir, los números de pin para las entradas y salidas son diferentes debido a que las conexiones de los pines de masa y alimentación son distintos. Para el 74LS83A, V_{CC} es el pin 5 y tierra

es el pin 12 en el encapsulado de 16 pines. Para el 74LS283, V_{CC} es el pin 16 y tierra es el pin 8, que es una configuración más estándar. Los diagramas de los pines y los símbolos lógicos de estos dos dispositivos se muestran en la Figura 6.10, en la que se indica la numeración de los pines sobre los símbolos lógicos.



(a) Diagramas de pines (b) 74LS83A (c) 74LS283

FIGURA 6.10
Sumadores MSI en paralelo de 4 bits.

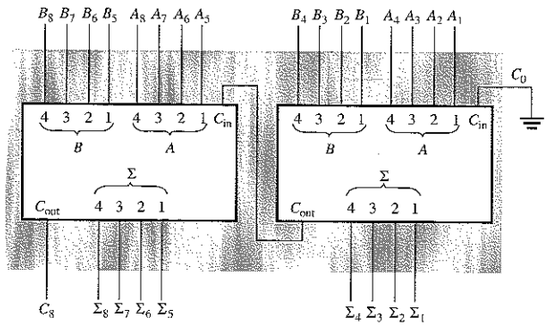
Parámetros temporales de una hoja de características. Recuérdese que las puertas lógicas tienen un retardo de propagación especificado, t_P, desde una entrada a la salida. Para los dispositivos lógicos MSI, existen varias especificaciones diferentes para este parámetro. El sumador paralelo de 4 bits dispone de 4 especificaciones para t_P, como se muestra en la Figura 6.11, tabla que es parte de una hoja de características del 74LS283.

Símbolo	Parámetro	Límites			Unidad
		Mínimo	Típico	Máximo	
t_{PLH} t_{PHL}	Propagation delay, C_0 input to any Σ output		16 15	24 24	ns
t_{PLH} t_{PHL}	Propagation delay, any A or B input to Σ outputs		15 15	24 24	ns
t_{PLH} t_{PHL}	Propagation delay, C_0 input to C_4 output		11 11	17 22	ns
t_{PLH} t_{PHL}	Propagation delay, any A or B input to C_4 output		11 12	17 17	ns

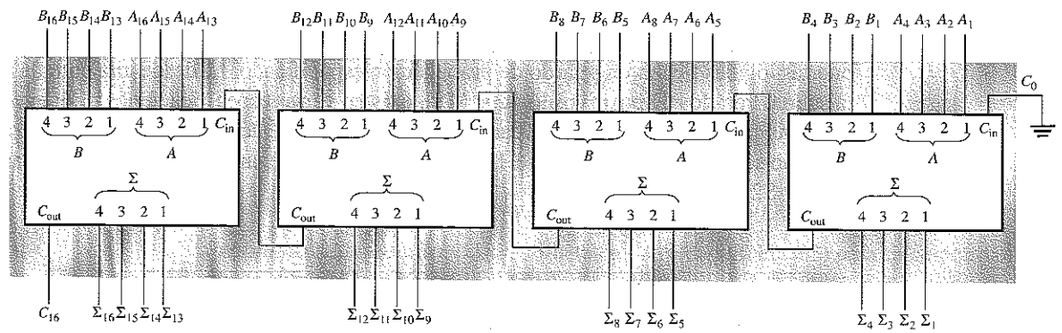
FIGURA 6.11
Características temporales para el 74LS283.

Expansión de sumadores

Un sumador en paralelo de 4 bits se puede expandir para realizar sumas de dos números de 8 bits, utilizando dos sumadores de cuatro bits y conectando la entrada de acarreo del sumador de menor orden (C_0) a tierra, ya que no existe acarreo en la posición del bit menos significativo, y la salida de acarreo del sumador de menor orden se conecta a la entrada de acarreo del sumador de orden superior, como se muestra en la Figura 6.12(a). Este proceso se denomina **conexión en cascada**. Obsérvese que, en este caso, el acarreo de salida se designa como C_8, dado que se genera a partir del bit que se encuentra en la posición número ocho. El sumador de menor orden es el que realiza la suma de los cuatro bits menos significativos, mientras que el sumador de orden superior es el que suma los cuatro bits más significativos de los dos números binarios de 8 bits.



(a) Sumadores de 4 bits conectados en cascada que forman un sumador de 8 bits



(b) Sumadores de 4 bits conectados en cascada que forman un sumador de 16 bits

FIGURA 6.12
Ejemplos de expansión de sumadores.

De forma similar, se pueden emplear cuatro sumadores de 4 bits en cascada para sumar dos números de 16 bits, como se muestra en la Figura 6.12(b). Obsérvese que el acarreo de salida se designa como C_{16}, ya que se genera a partir del bit que se encuentra en la posición dieciséis.

EJEMPLO 6.4 Demostrar cómo dos sumadores 74LS83A se pueden conectar para formar un sumador en paralelo de 8 bits. Hallar los bits de salida utilizando como entrada los siguientes números de 8 bits:

$$A_8A_7A_6A_5A_4A_3A_2A_1 = 10111001 \quad \text{y} \quad B_8B_7B_6B_5B_4B_3B_2B_1 = 10011110$$

Solución. Se utilizan dos sumadores paralelo de 4 bits 74LS83A para implementar el sumador de 8 bits. La única conexión entre los dos 74LS83A es la que une la salida de acarreo (pin 14) del sumador de menor orden a la entrada de acarreo (pin 13) del sumador de orden superior, como se muestra en la Figura 6.13. El pin 13 del sumador de orden inferior se conecta a masa (no hay entrada de acarreo).

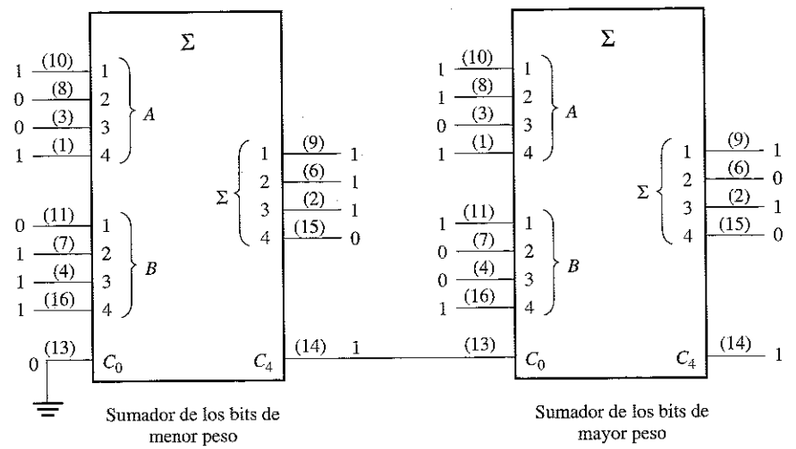


FIGURA 6.13
Dos sumadores 74LS83A conectados como un sumador en paralelo de 8 bits (la numeración de los pines se indica entre paréntesis).

Problema relacionado. Utilizar sumadores 74LS283 para implementar un sumador en paralelo de 12 bits.

Ejemplo de aplicación

Un ejemplo de aplicación de un sumador completo y de un sumador en paralelo es un sencillo sistema de recuento de votos, que se puede utilizar simultáneamente para proporcionar el número de votos afirmativos y el de negativos. Por ejemplo, este tipo de sistema podría utilizarse en una reunión de personas donde se necesite determinar de forma inmediata el número de opiniones favorables para tomar decisiones, o votar sobre determinados temas.

En su forma más sencilla, el sistema se compone de un interruptor para seleccionar las dos posibles opciones (afirmativa o negativa) de cada persona en la reunión y un display digital para mostrar el número de votos afirmativos, y otro para los negativos. El sistema básico se muestra en la Figura 6.14, para un número de personas

igual a 6, pero se puede ampliar a cualquier número de votantes, añadiendo nuevos módulos de 6 interruptores, sumadores en paralelo y circuitos de displays adicionales.

En la Figura 6.14, cada sumador completo puede generar la suma de hasta tres votos. La suma y el acarreo de salida de cada sumador completo se conectan a las dos entradas de menor orden de un sumador binario en paralelo. Las dos entradas de orden superior del sumador en paralelo se conectan a tierra (0), ya que nunca existe la posibilidad de que la entrada binaria sea mayor que 0011 (3 decimal). Para este sistema básico de 6 posiciones, las salidas del sumador en paralelo se conectan a un decodificador BCD a 7 segmentos que controla el display. Como ya se ha mencionado, se tienen que añadir circuitos adicionales si se decide ampliar el sistema.

Las resistencias entre las entradas de cada sumador completo y tierra aseguran que cada entrada se encuentra a nivel BAJO cuando el interruptor está en su posición neutra (se utiliza lógica CMOS). Cuando un interruptor se mueve a la posición "sí" o a la posición "no", se aplica un nivel de tensión ALTO (V_{CC}) a la entrada del sumador completo asociado.

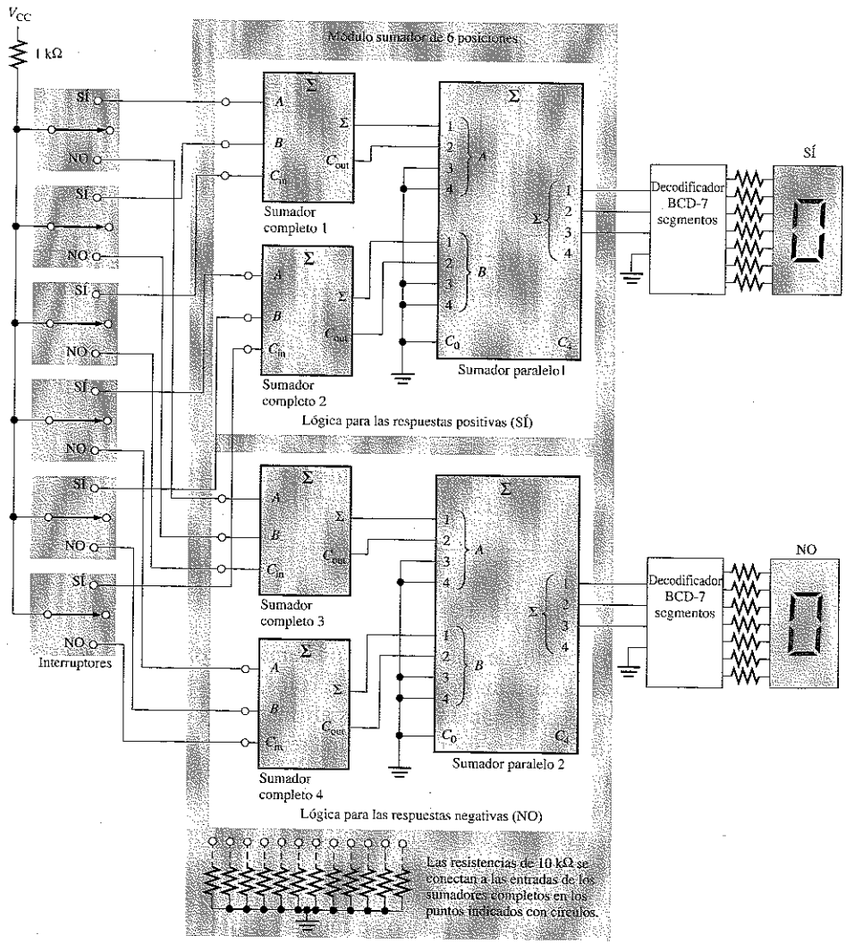


FIGURA 6.14
Sistema de votación realizado a partir de sumadores completos y sumadores binarios en paralelo.

REVISIÓN DE LA SECCIÓN 6.2

1. Se aplican dos números binarios de 4 bits (1101 y 1011) a un sumador en paralelo de 4 bits. El acarreo de entrada es 1. Determinar la suma (Σ) y el acarreo de salida.
2. ¿Cuántos sumadores 74LS283 se necesitarán para sumar dos números binarios, que representan números decimales iguales o inferiores a 1000_{10}?

6.3 ■ COMPARADORES

La función básica de un comparador consiste en comparar las magnitudes de dos cantidades binarias para determinar su relación. En su forma más sencilla, un circuito comparador determina si dos números son iguales.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Utilizar una puerta OR-exclusiva como comparador básico. ■ Analizar la lógica interna de un comparador de magnitud que posee tanto salida de igualdad como de desigualdad. ■ Utilizar el comparador de magnitud 74HC85 para la comparación de dos números binarios de 4 bits. ■ Conectar en cascada comparadores 74HC85, para comparar números de 8 o más bits.

Igualdad

Como ya vimos en el Capítulo 3, la puerta OR-exclusiva se puede emplear como un comparador básico, ya que su salida es 1 si sus dos bits de entrada son diferentes y 0 si son iguales. La Figura 6.15 muestra una puerta OR-exclusiva utilizada como comparador de 2 bits.

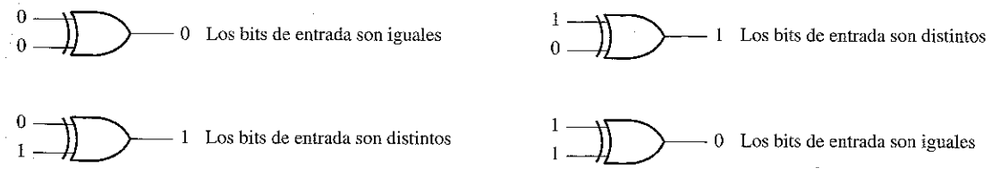


FIGURA 6.15
Funcionamiento del comparador básico.

Para comparar números binarios de dos bits, se necesita una puerta OR-exclusiva adicional. Los dos bits menos significativos (LSB) de ambos números se comparan mediante la puerta G_1 y los dos más significativos (MSB) son comparados mediante la puerta G_2, como se muestra en la Figura 6.16. Si los dos números son iguales, sus correspondientes bits también lo son, y la salida de cada puerta OR-exclusiva será 0. Si los correspondientes conjuntos de bits no son idénticos, la salida de la puerta OR-exclusiva será un 1.

Para obtener un único resultado de salida que indique la igualdad o desigualdad entre los dos números, se pueden usar dos inversores y una puerta AND, como mues-

tra la Figura 6.16. La salida de cada puerta OR-exclusiva se invierte y se aplica a la entrada de la puerta AND. Cuando los bits de entrada de cada OR-exclusiva son iguales, lo que quiere decir que los bits de ambos números son iguales, las entradas de la puerta AND son 1, por lo que el resultado a su salida también será 1. Cuando los dos números no son iguales, al menos uno o ambos conjuntos de bits será distinto, lo que da lugar a, al menos, un 0 en una de las entradas de la puerta AND, y el resultado a su salida será 0. Por tanto, la salida de la puerta AND indica la igualdad (1) o desigualdad (0) entre dos números.

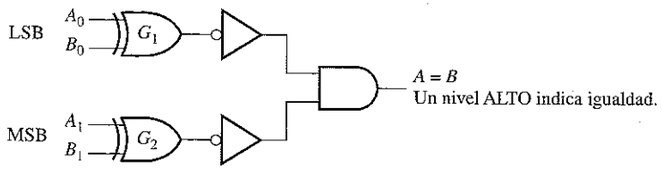


FIGURA 6.16
Diagrama lógico de la comparación de dos números de 2 bits.

El Ejemplo 6.5 ilustra esta operación para dos casos específicos. La puerta OR-exclusiva y el inversor se han reemplazado por un símbolo NOR-exclusiva. Recordemos que en la aplicación desarrollada en el Capítulo 3, utilizamos este tipo de circuito para comparar dos números de 8 bits.

EJEMPLO 6.5 Aplicar cada uno de los siguientes conjuntos de números binarios a las entradas del comparador de la Figura 6.17 y determinar la salida, evaluando los niveles lógicos a través del circuito.

(a) 10 y 10 **(b)** 11 y 10

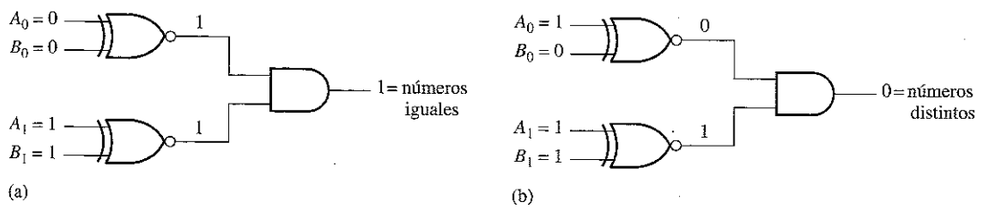


FIGURA 6.17

Solución

(a) La salida es **1** para las entradas 10 y 10, como se ve en la Figura 6.17(a).
(b) La salida es **0** para las entradas 11 y 10, como se ve en la Figura 6.17(b).

Problema relacionado. Repetir el proceso para las entradas binarias 01 y 10.

NOTAS INFORMÁTICAS
En una computadora, la *caché* es una memoria intermedia muy rápida entre la unidad de procesamiento central (CPU) y la memoria principal, más lenta. La CPU solicita los datos enviando la *dirección* (ubicación unívoca) en memoria. Parte de esta dirección se denomina *marcador*. El *comparador de marcadores de dirección* compara el marcador de la CPU con el marcador del directorio de la caché. Si ambos son iguales, quiere decir que los datos direccionados se encuentran ya en la memoria y se recuperan de forma muy rápida. Si los marcadores son distintos, los datos deben recuperarse de la memoria principal a una velocidad mucho más lenta.

Como ya se ha visto en el Capítulo 3, un circuito comparador básico se puede ampliar para poder tratar cualquier número de bits. La puerta AND establece la condición de que todos los bits de los dos números que se comparan tienen que ser iguales si los números lo son.

Desigualdad

Además de disponer de una salida que indica si los dos números son iguales, muchos circuitos integrados comparadores tienen salidas adicionales que indican cuál de los dos números que se comparan es el mayor. Esto significa que existe una salida que indica cuándo el número A es mayor que el número B ($A > B$) y otra salida que indica cuándo A es menor que B ($A < B$), como se muestra en el símbolo lógico del comparador de cuatro bits de la Figura 6.18.

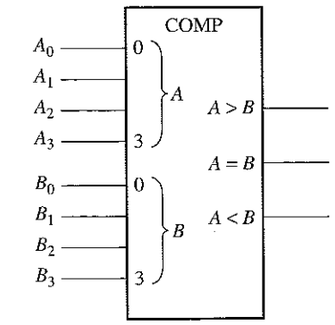


FIGURA 6.18
Símbolo lógico para un comparador de 4 bits con indicación de desigualdad.

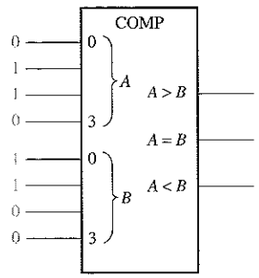
Para determinar una desigualdad entre los números binarios A y B, en primer lugar se examina el bit de mayor orden de cada número. Las posibles condiciones son las siguientes:

1. Si $A_3 = 1$ y $B_3 = 0$, entonces A es mayor que B.
2. Si $A_3 = 0$ y $B_3 = 1$, entonces A es menor que B.
3. Si $A_3 = B_3$, entonces tenemos que examinar los siguientes bits de orden inmediatamente inferior.

Estas tres proposiciones son válidas para cada posición que ocupen los bits dentro del número. El procedimiento general consiste en comprobar una desigualdad en cualquier posición, comenzando por los bits más significativos (MSB). Cuando se encuentra una desigualdad, la relación entre ambos números queda establecida y cualquier otra desigualdad entre bits con posiciones de orden menor debe ignorarse, ya que podrían indicar una relación entre los números completamente opuesta. *La relación de más alto orden es la que tiene prioridad.*

EJEMPLO 6.6 Determinar las salidas $A = B$, $A > B$ y $A < B$ para los números de entrada mostrados en el comparador de la Figura 6.19.

FIGURA 6.19



Solución. El número que hay en las entradas A es 0110 y el número que hay en las entradas B es 0011. La **salida $A > B$ está a nivel ALTO y las restantes salidas están a nivel BAJO.**

Problema relacionado. ¿Cuáles serán las salidas del comparador cuando $A_3A_2A_1A_0 = 1001$ y $B_3B_2B_1B_0 = 1010$?

Un comparador de magnitud de 4 bits MSI

El 74HC85 es un comparador de tipo MSI, que también se encuentra disponible en otras familias de circuitos integrados. El diagrama de pines y el símbolo lógico se muestran en la Figura 6.20. Obsérvese que este dispositivo tiene todas las entradas y salidas del comparador visto anteriormente y, además, tiene tres entradas en cascada: $A < B$, $A = B$ y $A > B$. Estas entradas permiten utilizar varios comparadores en cascada para la comparación de cualquier número binario con más de cuatro bits. Para expandir el comparador, las salidas $A < B$, $A = B$ y $A > B$ del comparador de menor orden se conectan en cascada a las entradas del siguiente comparador de orden inmediatamente superior. El comparador de menor orden tiene que tener un nivel ALTO en la entrada $A = B$ y un nivel BAJO en las entradas $A > B$ y $A < B$.

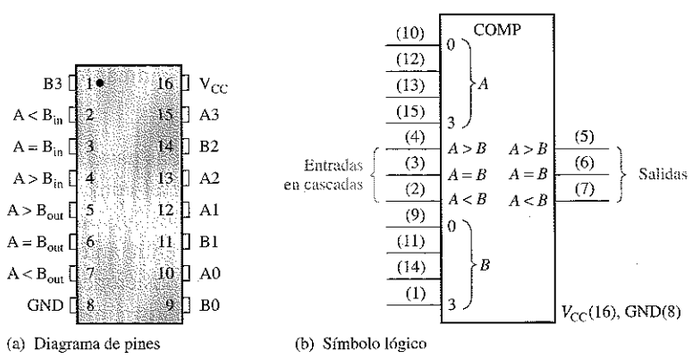


FIGURA 6.20
Diagrama de pines y símbolo lógico del comparador de 4 bits de magnitud 74HC85 (la numeración de los pines se muestra entre paréntesis).

EJEMPLO 6.7 Utilizar comparadores 74HC85 para comparar las magnitudes de dos números de 8 bits. Dibujar los comparadores con sus correspondientes interconexiones.

Solución. Se necesitan dos 74HC85 para comparar dos números de 8 bits. Éstos se conectan en cascada como se muestra en la Figura 6.21, empleando una disposición en cascada.

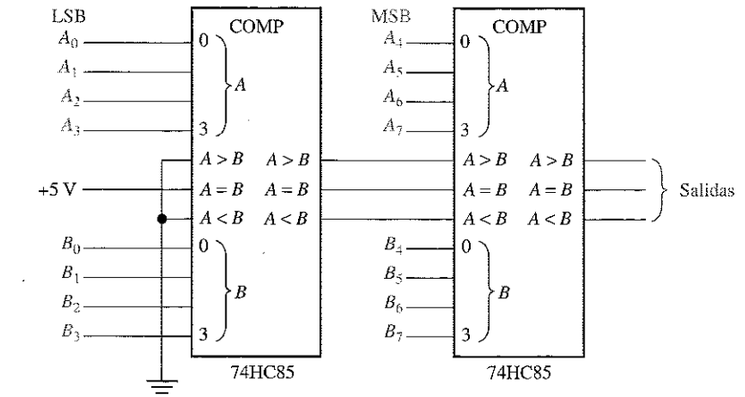


FIGURA 6.21
Comparador de 8 bits formado por dos 74HC85.

Problema relacionado. Ampliar el circuito de la Figura 6.21 para realizar un comparador de 16 bits.

Consejos Prácticos

La mayoría de los dispositivos CMOS contienen circuitería de protección para protegerse frente a daños generados por altas tensiones estáticas o campos eléctricos. Sin embargo, deben tomarse precauciones para evitar la aplicación de cualquier tensión mayor que la tensión máxima nominal. Para un funcionamiento correcto, las tensiones de entrada y salida deberían estar comprendidas entre tierra y V_{CC}. Recuérdese también que las entradas no utilizadas deberán conectarse siempre a un nivel lógico apropiado (tierra o V_{CC}). Las salidas no utilizadas pueden dejarse en circuito abierto.

REVISIÓN DE LA SECCIÓN 6.3

1. Los números binarios $A = 1011$ y $B = 1010$ se aplican a las entradas de un 74HC85. Determinar las salidas.
2. Los números binarios $A = 11001011$ y $B = 11010100$ se aplican al comparador de 8 bits de la Figura 6.21. Determinar el estado de los pines de salida 5, 6 y 7 en cada uno de los 74HC85.

6.4 ■ DECODIFICADORES

La función básica de un decodificador es detectar la presencia de una determinada combinación de bits (código) en sus entradas y señalar la presencia de este código mediante un cierto nivel de salida. En su forma más general, un decodificador posee, líneas de entrada para gestionar n bits, y en una de las 2^n líneas de salida indica la presencia de una o más combinaciones de n bits. En esta sección, se presentarán varios tipos de decodificadores. Los principios básicos se pueden extender a otros tipos de decodificadores.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Definir *decodificador*. ■ Diseñar un circuito lógico para decodificar cualquier combinación de bits. ■ Describir el decodificador binario-decimal 74HC154. ■ Describir el decodificador BCD-decimal 74HC42. ■ Ampliar los decodificadores para poder tratar códigos con un número de bits superior. ■ Describir el decodificador BCD a 7 segmentos 74LS47. ■ Explicar la supresión de cero en los displays de 7 segmentos. ■ Utilizar los decodificadores en aplicaciones específicas.

NOTAS INFORMÁTICAS Una *instrucción* indica a la computadora qué operación debe realizar. Las instrucciones se especifican en código máquina (1s y 0s) y, para que la computadora ejecute una instrucción, ésta debe ser decodificada. La decodificación de las instrucciones es uno de los pasos en la *pipeline* (secuencia de procesamiento) de las instrucciones; los pasos de dicho proceso son: la instrucción se lee desde la memoria (extracción de la instrucción), la instrucción se decodifica, se leen los operandos desde la memoria (extracción de operandos), se ejecuta la instrucción y el resultado se escribe de nuevo en memoria. Básicamente, el procesamiento *pipeline* permite que se comience a procesar la siguiente instrucción antes de haber completado la instrucción actual.

El decodificador binario básico

Supongamos que necesitamos determinar cuándo aparece el número binario 1001 en las entradas de un circuito digital. Se puede utilizar una puerta AND como elemento básico de decodificación, ya que produce una salida a nivel ALTO sólo cuando todas sus entradas están a nivel ALTO. Por tanto, debe asegurarse de que todas las entradas de la puerta AND estén a nivel ALTO cuando se introduce el número 1001, lo cual se puede conseguir invirtiendo las dos entradas centrales (cuyos bits son 0), como se muestra en la Figura 6.22.

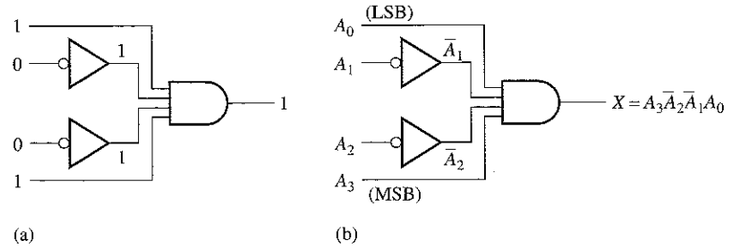


FIGURA 6.22
Lógica de decodificación del código binario 1001 con una salida activa a nivel ALTO.

La ecuación lógica para el **decodificador** de la Figura 6.22(a) se desarrolla como se ilustra en la Figura 6.22(b). Se debe comprobar que la salida es siempre 0 excepto cuando se aplican las entradas $A_0 = 1$, $A_1 = 0$, $A_2 = 0$ y $A_3 = 1$. A_0 es el bit menos significativo y A_3 el más significativo. *En este libro, cuando se representa un número binario o cualquier otro código de pesos, el bit menos significativo siempre es el situado más a la derecha cuando el número se escribe en sentido horizontal, y el de más arriba cuando se escribe en vertical, a menos que se indique lo contrario.*

Si se utiliza una puerta NAND en lugar de una AND, como se muestra en la Figura 6.22, una salida a nivel BAJO indicará la presencia del código binario adecuado, que en este caso es 1001.

EJEMPLO 6.8 Determinar la lógica requerida para decodificar el número binario 1011 de manera que produzca un nivel ALTO en la salida.

Solución. La función de decodificación la podemos realizar complementando sólo las variables cuyo valor es 0 en el número binario deseado, tal y como sigue:

$$X = A_3\overline{A}_2A_1A_0 \quad (1011)$$

Esta función puede implementarse conectando las variables verdaderas (no complementadas) A_0, A_1 y A_3 directamente a las entradas de la puerta AND, e invirtiendo la variable A_2 antes de aplicarla a la puerta AND. La lógica de decodificación se muestra en la Figura 6.23.

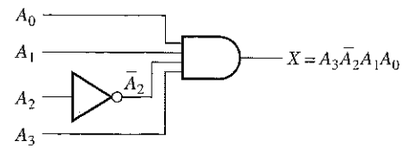


FIGURA 6.23
Lógica de decodificación para obtener una salida a nivel ALTO cuando se aplica el código 1011 en las entradas.

Problema relacionado. Desarrollar la lógica requerida para detectar el código binario 10010 y generar una salida activa a nivel BAJO.

El decodificador de 4 bits

Para poder decodificar todas las posibles combinaciones de cuatro bits, se necesitan 16 puertas de decodificación ($2^4 = 16$). Este tipo de decodificador se denomina comúnmente *decodificador de 4 líneas a 16 líneas*, ya que existen 4 entradas y 16 salidas, o también se le llama *decodificador 1 de 16*, ya que para cualquier código dado en las entradas, sólo se activa una de las dieciséis posibles salidas. En la Tabla 6.4 se muestra una lista de los 16 códigos binarios y sus correspondientes funciones de decodificación.

Si se necesita una salida activa a nivel BAJO para cada número decodificado, el decodificador completo se puede implementar mediante puertas NAND e inversores. Para decodificar cada uno de los 16 códigos binarios se requieren 16 puertas NAND (las puertas AND se pueden usar para producir salidas activas a nivel ALTO).

TABLA 6.4
Funciones de decodificación y tabla de verdad para un decodificador de 4 líneas a 16 líneas con salidas activas a nivel BAJO.

Dígito decimal	Entradas binarias A_3	A_2	A_1	A_0	Función de decodificación	Salidas 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	$\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	0	1	$\overline{A}_3\overline{A}_2\overline{A}_1A_0$	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	0	1	0	$\overline{A}_3\overline{A}_2A_1\overline{A}_0$	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
3	0	0	1	1	$\overline{A}_3\overline{A}_2A_1A_0$	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
4	0	1	0	0	$\overline{A}_3A_2\overline{A}_1\overline{A}_0$	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
5	0	1	0	1	$\overline{A}_3A_2\overline{A}_1A_0$	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
6	0	1	1	0	$\overline{A}_3A_2A_1\overline{A}_0$	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
7	0	1	1	1	$\overline{A}_3A_2A_1A_0$	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
8	1	0	0	0	$A_3\overline{A}_2\overline{A}_1\overline{A}_0$	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
9	1	0	0	1	$A_3\overline{A}_2\overline{A}_1A_0$	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
10	1	0	1	0	$A_3\overline{A}_2A_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
11	1	0	1	1	$A_3\overline{A}_2A_1A_0$	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
12	1	1	0	0	$A_3A_2\overline{A}_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
13	1	1	0	1	$A_3A_2\overline{A}_1A_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
14	1	1	1	0	$A_3A_2A_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
15	1	1	1	1	$A_3A_2A_1A_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

El símbolo lógico de un decodificador de 4 líneas a 16 líneas (1 de 16) con salidas activas a nivel BAJO se muestra en la Figura 6.24. La etiqueta BIN/DEC indica que una entrada binaria produce su correspondiente salida decimal. Las etiquetas 8, 4, 2 y 1 en las entradas representan los pesos binarios de los bits de entrada (2^3 2^2 2^1 2^0).

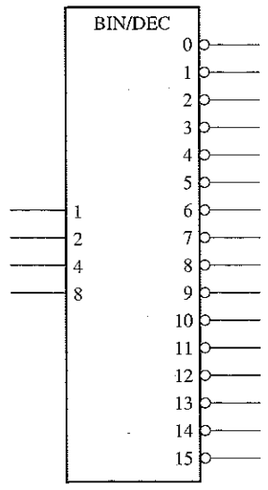
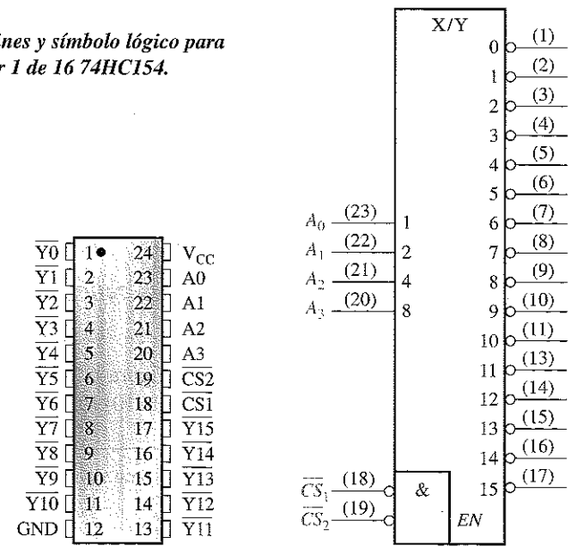


FIGURA 6.24
Símbolo lógico de un decodificador de 4 líneas a 16 líneas (1 de 16).

Un decodificador MSI 1 de 16

El 74HC154 es un buen ejemplo de un decodificador MSI. Su símbolo lógico se muestra en la Figura 6.25. En este tipo de dispositivo existe una función de activación (enable, *EN*), que se implementa mediante una puerta NOR utilizada como negativa-AND. En las entradas de selección del chip, $\overline{CS}_1$ y $\overline{CS}_2$, se requiere un nivel BAJO para obtener en la salida de la puerta de activación (EN, *enable*) un nivel ALTO. La salida de la puerta de activación se conecta a una entrada de *cada* puerta NAND del decodificador, por lo que debe estar a nivel ALTO para que las puertas NAND se activen. Si la puerta de activación no se activa mediante un nivel BAJO en ambas entradas, entonces las 16 salidas (Y) del decodificador estarán a nivel ALTO independientemente del estado de las cuatro variables de entrada A_0, A_1, A_2 y A_3.

FIGURA 6.25
Diagrama de pines y símbolo lógico para el decodificador 1 de 16 74HC154.



(a) Diagrama de pines (b) Símbolo lógico

EJEMPLO 6.9 Una cierta aplicación requiere decodificar un número de 5 bits. Utilizar decodificadores 74HC154 para implementar el circuito lógico. El número binario se representa de la forma $A_4A_3A_2A_1A_0$.

Solución. Dado que el 74HC154 puede procesar únicamente cuatro bits, habrá que usar dos decodificadores para los cinco bits. El quinto bit, A_4, está conectado a las entradas de selección del chip $\overline{CS}_1$ y $\overline{CS}_2$, de uno de los decodificadores y $\overline{A}_4$ se conecta a las entradas de activación del otro decodificador, como se muestra en la Figura 6.26. Cuando el número decimal es 15 o menor, $A_4 = 0$, se activa el decodificador de menor orden y se desactiva el de mayor orden. Cuando el número decimal es mayor que 15, $A_4 = 1$, luego $\overline{A}_4 = 0$, lo que hace que se active el decodificador de orden superior y se desactive el de orden inferior.

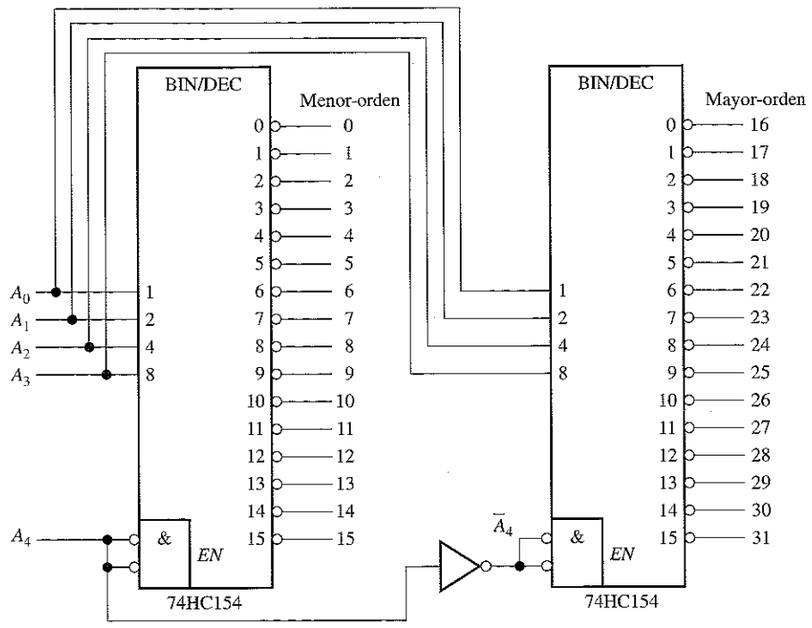


FIGURA 6.26
Decodificador de 5 bits construido con dos 74HC154.

Problema relacionado. En la Figura 6.26, determinar la salida que se activa al introducir el código binario de entrada 10110.

Ejemplo de aplicación

Los decodificadores se utilizan en muchos tipos de aplicaciones. Un ejemplo es la selección de entradas y salidas en las computadoras, como se muestra en el diagrama general de la Figura 6.27.

Las computadoras se tienen que comunicar con una gran variedad de dispositivos externos, denominados *periféricos*, enviando y/o recibiendo datos a través de lo que se conoce como puertos de entrada/salida (E/S). Estos dispositivos externos incluyen impresoras, modems, escáneres, unidades de disco externas, teclados, monitores y otras computadoras. Como se indica en la Figura 6.27, se emplea un decodificador para seleccionar el puerto de E/S determinado por la computadora, de forma que los datos puedan ser enviados o recibidos desde algún dispositivo externo concreto.

Cada puerto de E/S tiene un número, denominado dirección, que lo identifica unívocamente. Cuando la computadora desea comunicarse con algún dispositivo en particular, envía el código de dirección apropiado del puerto de E/S al que está conectado el dispositivo en cuestión. Esta dirección binaria del puerto se decodifica, activándose la salida del decodificador apropiada que habilita el correspondiente puerto de E/S.

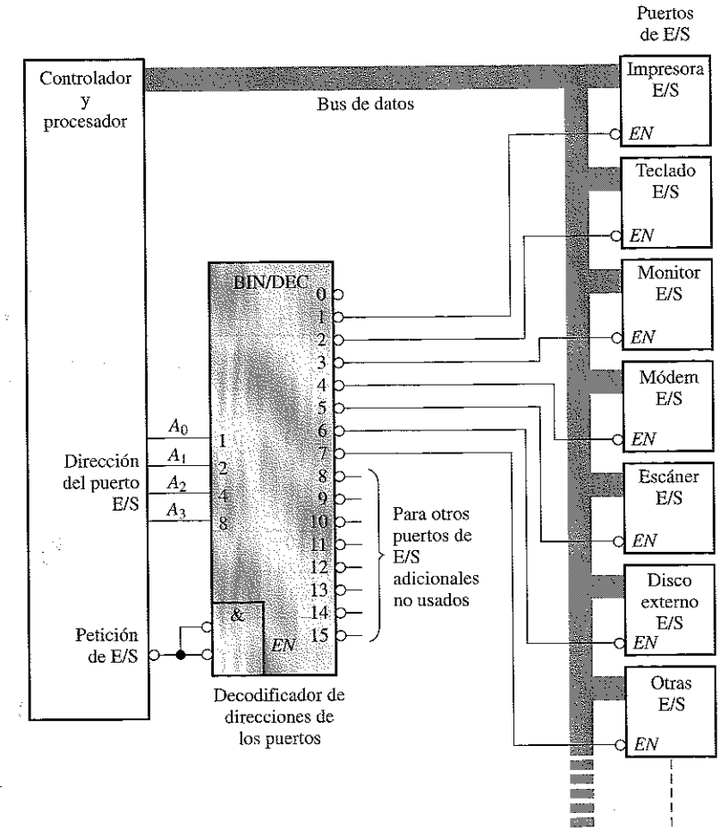


FIGURA 6.27
Un sistema simplificado de puertos de E/S con un decodificador de direcciones de puerto con sólo cuatro líneas de dirección.

Como se muestra en la Figura 6.27, los datos binarios se transfieren dentro de la computadora a través de un bus de datos, que consiste en un conjunto de líneas paralelas. Por ejemplo, un bus de 8 bits consta de ocho líneas paralelas que pueden transmitir un byte de datos de una sola vez. El bus de datos está conectado a todos los puertos de E/S, pero los datos que son recibidos o transmitidos sólo pasarán a través del puerto que se encuentre activado por el decodificador de direcciones de puertos.

El decodificador BCD a decimal

Un decodificador BCD a decimal convierte cada código BCD (código 8421) en uno de los diez posibles dígitos decimales. Frecuentemente, se le denomina *decodificador de 4 líneas a 10 líneas* o *decodificador 1 de 10*.

El método de implementación es el mismo que hemos visto anteriormente para el decodificador de 4 líneas a 16 líneas, excepto que ahora sólo se requieren 10 puer-

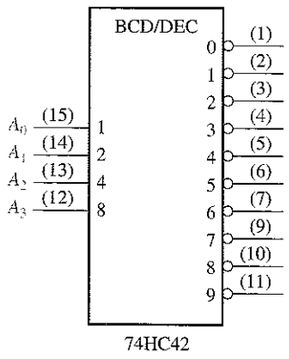
tas decodificadoras, dado que el código BCD sólo representa los diez dígitos decimales de 0 a 9. En la Tabla 6.5 se muestra una lista de los diez códigos BCD y sus correspondiente funciones de decodificación. Cada una de estas funciones se implementa mediante puertas NAND para proporcionar salidas activas a nivel BAJO. Si se requirieran salidas activas a nivel ALTO, se utilizarían puertas AND para la decodificación. La lógica es idéntica a la de las diez primeras puertas del decodificador de 4-líneas a 16 líneas (véase la Tabla 6.4)

TABLA 6.5
Funciones de decodificación BCD.

Dígito decimal	Código BCD A_3	A_2	A_1	A_0	Función de decodificación
0	0	0	0	0	$\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$
1	0	0	0	1	$\overline{A}_3\overline{A}_2\overline{A}_1A_0$
2	0	0	1	0	$\overline{A}_3\overline{A}_2A_1\overline{A}_0$
3	0	0	1	1	$\overline{A}_3\overline{A}_2A_1A_0$
4	0	1	0	0	$\overline{A}_3A_2\overline{A}_1\overline{A}_0$
5	0	1	0	1	$\overline{A}_3A_2\overline{A}_1A_0$
6	0	1	1	0	$\overline{A}_3A_2A_1\overline{A}_0$
7	0	1	1	1	$\overline{A}_3A_2A_1A_0$
8	1	0	0	0	$A_3\overline{A}_2\overline{A}_1\overline{A}_0$
9	1	0	0	1	$A_3\overline{A}_2\overline{A}_1A_0$

EJEMPLO 6.10 El 74HC42 es un circuito integrado decodificador BCD-decimal. Su símbolo lógico se muestra en la Figura 6.28. Dibujar las señales de salida si se aplican las formas de onda de entrada de la Figura 6.29(a) a las entradas del 74HC42.

FIGURA 6.28
El decodificador BCD a decimal 74HC42.



Solución. Las formas de onda de salida se muestran en la Figura 6.29(b). Como puede verse, las entradas al decodificador constituyen una secuencia de dígitos de 0 a 9. Las formas de onda de salida del cronograma indican dicha secuencia.

Problema relacionado. Construir un diagrama de tiempos que muestre las señales de entrada y de salida para el caso en que la secuencia binaria de entrada origine los siguientes números decimales: 0, 2, 4, 6, 8, 1, 3, 5 y 9.

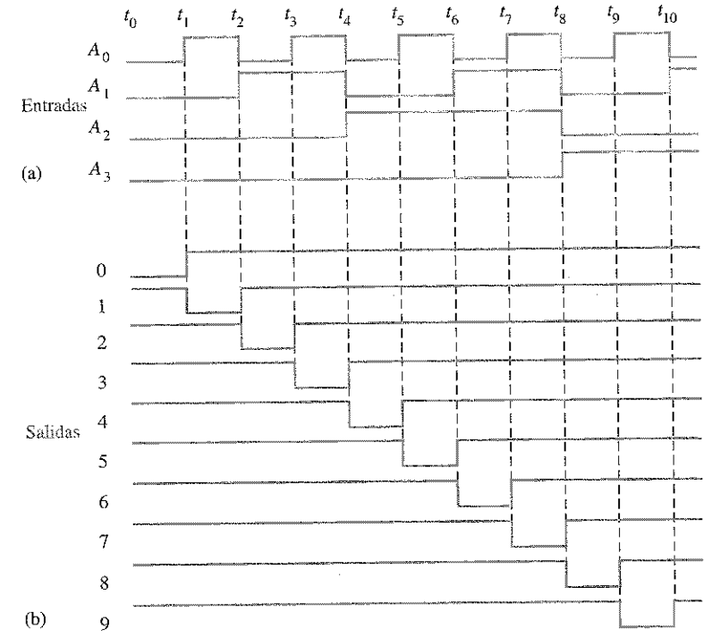


FIGURA 6.29

El decodificador BCD a 7 segmentos

Como se ha visto en la aplicación del Capítulo 4, este tipo de decodificador acepta código BCD en sus entradas y proporciona salidas capaces de excitar un display de 7 segmentos para indicar un dígito decimal. En la Figura 6.30 se muestra el diagrama lógico de un decodificador básico de 7 segmentos.

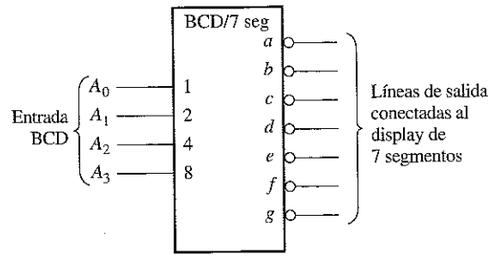
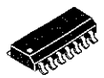


FIGURA 6.30
Símbolo lógico de un decodificador/controlador BCD a 7 segmentos con salidas activas a nivel BAJO.



Un decodificador /controlador BCD a 7 segmentos MSI

El 74LS47 es un ejemplo de dispositivo MSI que decodifica una entrada BCD y controla un display de 7 segmentos. Además de estas características de decodificación y control, el 74LS47 posee características adicionales, como las indicadas en el símbolo lógico de la Figura 6.31 por las funciones $\overline{LT}$, $\overline{RBI}$ y $\overline{BI}/\overline{RBO}$. Como indican los círculos del símbolo lógico, todas las salidas (de *a* a *g*) son activas a nivel BAJO, al igual que lo son $\overline{LT}$ (lamp test), *RBI* (ripple blanking input) y $\overline{BI}/\overline{RBO}$ (blanking input/ripple blanking output). Las salidas pueden controlar directamente un display de 7 segmentos en ánodo común. Recuerde que los displays de 7 segmentos se trataron en el Capítulo 4.

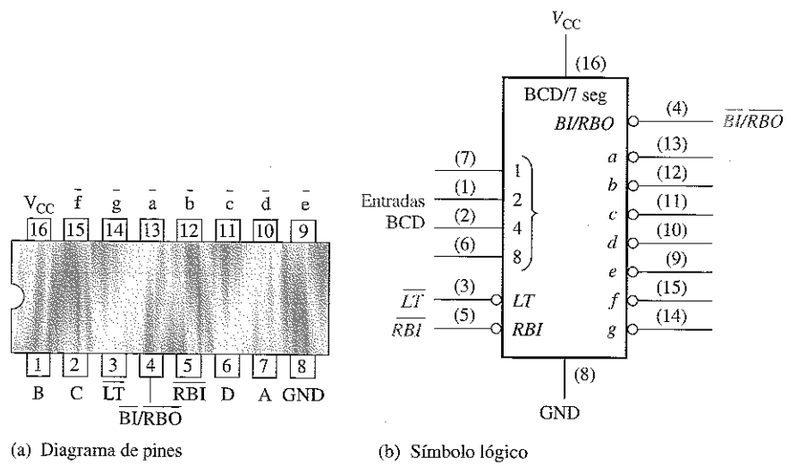


FIGURA 6.31
Diagrama de pines y símbolo lógico para el codificador/controlador BCD a 7 segmentos 74LS47.

Además de decodificar una entrada BCD y generar las apropiadas salidas para 7 segmentos, el 74LS47 posee las funciones de entrada de comprobación y de supresión de cero.

Entrada de comprobación. Cuando se aplica un nivel BAJO a la entrada $\overline{LT}$ y la entrada $\overline{BI}/\overline{RBO}$ está a nivel ALTO, se encienden todos los segmentos del display. La entrada de comprobación se utiliza para verificar que ninguno de los segmentos está fundido.

Supresión de cero. La **supresión de cero** es una característica utilizada en displays de varios dígitos para eliminar los ceros innecesarios. Por ejemplo, en un display de 6 dígitos, el número 6.4 podría mostrarse como 006.400 si no se eliminaran los ceros. La supresión de ceros al principio de un número recibe el nombre de *supresión anterior de cero*, mientras que si son los últimos los que se eliminan se denomina *supresión posterior de cero*. Téngase en cuenta que únicamente se eliminan los ceros que no son esenciales. Gracias a la supresión de cero, el número 030.080 se visualiza como 30.08 (los ceros esenciales permanecen).

La supresión de cero en el 74LS47 se logra utilizando las funciones $\overline{RBI}$ y $\overline{BI}/\overline{RBO}$. $\overline{RBI}$ es la entrada de borrado en cascada y $\overline{RBO}$ es la salida de borrado en cascada del 74LS47, las cuales se utilizan para la supresión de cero. $\overline{BI}$ es la entrada de borrado y comparte el mismo pin con $\overline{RBO}$. En otras palabras, el pin $\overline{BI}/\overline{RBO}$ puede utilizarse como salida o como entrada. Cuando se emplea como $\overline{BI}$ (entrada de borrado), todas las salidas están a nivel ALTO (segmentos desactivados) cuando $\overline{BI}$ está a nivel BAJO, anulando el resto de entradas. La función $\overline{BI}$ no forma parte de las características de supresión de cero del dispositivo.

Todas las salidas del decodificador correspondientes a los segmentos se encuentran inactivas (nivel ALTO) cuando se introduce el código cero (0000) en sus entradas BCD siempre que su $\overline{RBI}$ está a nivel BAJO. Esto origina que el display no muestre nada y que la salida $\overline{RBO}$ esté a nivel BAJO.

El diagrama lógico de la Figura 6.32(a) ilustra la supresión anterior de cero para un número entero. El dígito más significativo (el de más a la izquierda) desaparece siempre que se introduzca el código del 0 en sus entradas BCD, ya que la $\overline{RBI}$ del decodificador de orden más alto se conecta a tierra de forma que siempre esté a nivel BAJO. La salida $\overline{RBO}$ de cada decodificador se conecta a la entrada $\overline{RBI}$ del siguiente decodificador de menor orden, de manera que se eliminan todos los ceros a la izquierda del primer dígito que no sea cero. Por ejemplo, en la parte (a) de la figura, los dos dígitos más significativos son cero y, por tanto, se eliminan. Los dos dígitos restantes, 3 y 9, se visualizan en el display.

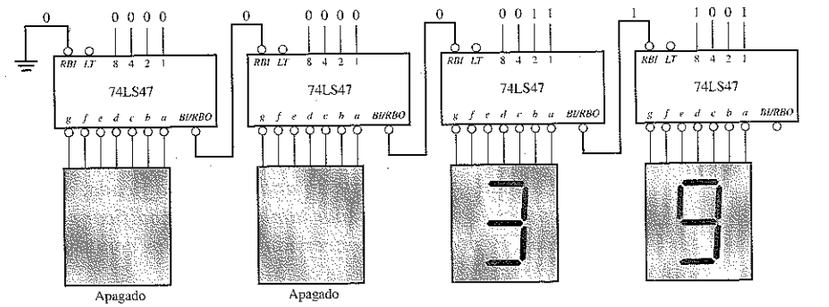


FIGURA 6.32
Ejemplos de supresión de cero mediante un decodificador/controlador BCD a 7 segmentos 74LS47.

El diagrama lógico de la Figura 6.32(b) ilustra la supresión posterior de cero para un número fraccionario. El dígito de menor orden (el de más a la derecha) se suprime siempre que se introduzca un código nulo en sus entradas BCD, ya que la entrada $\overline{RBI}$ está conectada a masa. La salida $\overline{RBO}$ de cada decodificador está conectada a la entrada $\overline{RBI}$ del decodificador siguiente de orden superior, de forma que todos los ceros a la derecha del primer dígito no nulo son eliminados. En la parte (b) de la figura, los dos dígitos menos significativos son cero, luego se eliminan. Los dos dígitos restantes, 5 y 7, se muestran en el display. Para combinar la supresión anterior de cero con la posterior en un mismo display y, además, tener posibilidad de introducir puntos decimales, necesitamos circuitos lógicos adicionales.

REVISIÓN DE LA SECCIÓN 6.4

1. Un decodificador de 3 líneas a 8 líneas se puede utilizar como decodificador octal a decimal. Cuando se introduce el número binario 101 en sus entradas, ¿qué línea de salida se activa?
2. ¿Cuántos decodificadores 1 de 16 74HC154 son necesarios para decodificar un número binario de 6 bits?
3. ¿Qué elegiríamos para controlar un display de 7 segmentos con cátodo común, un decodificador/controlador con salidas activas a nivel BAJO o con salidas activas a nivel ALTO?

6.5 ■ CODIFICADORES

Un codificador es un circuito lógico combinacional que, esencialmente, realiza la función "inversa" del decodificador. Un codificador permite que se introduzca en una de sus entradas un nivel activo que representa un dígito, como puede ser un dígito decimal u octal, y lo convierte en una salida codificada, como BCD o binario. Los codificadores se pueden diseñar también para codificar símbolos diversos y caracteres alfabéticos. El proceso de conversión de símbolos comunes o números a un formato codificado recibe el nombre de codificación.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Determinar la lógica de un codificador decimal. ■ Explicar la finalidad de la característica de prioridad en los codificadores. ■ Describir el codificador de prioridad decimal a BCD 74HC147. ■ Describir el codificador de prioridad octal a binario 74F148. ■ Ampliar un codificador. ■ Utilizar los codificadores en aplicaciones específicas.

Codificador decimal-BCD

Este tipo de **codificador** posee diez entradas, una para cada dígito decimal, y cuatro salidas que corresponden al código BCD, como se muestra en la Figura 6.33. Este es un codificador básico de 10 líneas a 4 líneas.

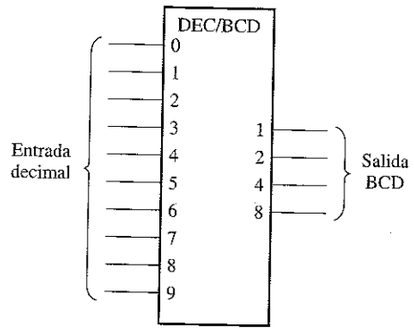
El código BCD (8421) se muestra en la Tabla 6.6. A partir de esta tabla podemos determinar la relación entre cada bit BCD y los dígitos decimales, con el fin de analizar la lógica. Por ejemplo, el bit más significativo del código BCD, A_3, es siempre un 1 para los dígitos decimales 8 o 9. La expresión OR para el bit A_3 en función de los dígitos decimales puede por tanto escribirse como:

$$A_3 = 8 + 9$$

NOTAS INFORMÁTICAS

Se puede pensar en un *ensamblador* como en un codificador de software, ya que interpreta las instrucciones mnemónicas con las que se ha redactado un programa y lleva a cabo la codificación necesaria para convertir cada mnemónico en una instrucción de código máquina (series de 1s y 0s), que la computadora puede entender. Ejemplos de mnemónicos de instrucciones son ADD, MOV (move data, desplazamiento de datos), MUL (multiplicación), XOR, JMP (jump, salto) y OUT (salida a un puerto).

FIGURA 6.33
Símbolo lógico de un codificador decimal a BCD.



El bit A_2 es siempre un 1 para los dígitos decimales 4, 5, 6 o 7 y puede expresarse como una función OR de la manera siguiente:

$$A_2 = 4 + 5 + 6 + 7$$

El bit A_1 es siempre un 1 para los dígitos decimales 2, 3, 6 o 7 y puede expresarse como:

$$A_1 = 2 + 3 + 6 + 7$$

Finalmente, A_0 es siempre un 1 para los dígitos 1, 3, 5, 7 o 9. La expresión para A_0 es:

$$A_0 = 1 + 3 + 5 + 7 + 9$$

TABLA 6.6

Dígito decimal	Código BCD A_3	A_2	A_1	A_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Ahora vamos a implementar el circuito lógico necesario para codificar en código BCD cada dígito decimal, utilizando las expresiones lógicas que se acaban de desarrollar. Consiste simplemente en aplicar la operación OR a los dígitos decimales de

entrada apropiados, para así formar cada salida BCD. La lógica del codificador que resulta de estas expresiones se muestra en la Figura 6.34.

El funcionamiento básico del circuito de la Figura 6.34 es el siguiente: cuando aparece un nivel ALTO en *una* de las líneas de entrada correspondientes a los dígitos decimales, se generan los niveles apropiados en las cuatro líneas BCD de salida. Por ejemplo, si la línea de entrada 9 está a nivel ALTO (suponiendo que todas las demás entradas están a nivel BAJO), esta condición producirá un nivel ALTO en las salidas A_0 y A_3, y un nivel BAJO en A_1 y A_2, que es el código BCD (1001) del número decimal 9.

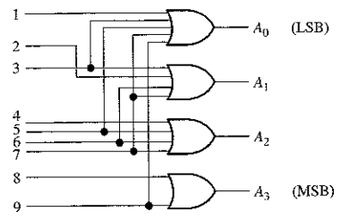


FIGURA 6.34
Diagrama lógico básico de un codificador decimal-BCD. No se necesita una entrada para el dígito 0, ya que las salidas BCD están todas a nivel BAJO cuando no hay entradas a nivel ALTO.

Codificador con prioridad decimal a BCD. Este tipo de codificador realiza la misma función de codificación básica que hemos visto anteriormente. Además, ofrece una flexibilidad adicional en lo relativo a que puede utilizarse en aplicaciones que requieren detección de prioridad. La función de prioridad significa que el codificador producirá una salida BCD correspondiente al *dígito decimal de entrada de más alto orden* que se encuentre activo, e ignorará cualquier otra entrada de menor orden que esté activa. Por ejemplo, si las entradas 6 y 3 se encuentran activas, la salida BCD será 0110 (que representa al número decimal 6).

Codificador MSI decimal-BCD

El 74HC147 es un **codificador con prioridad** con entradas activas a nivel BAJO (0) para los dígitos decimales del 1 al 9, y salidas BCD activas a nivel BAJO, como se indica en el símbolo lógico de la Figura 6.35. Una salida BCD cero se consigue cuando ninguna de las entradas está activa. La numeración de los pines del dispositivo se muestra entre paréntesis.

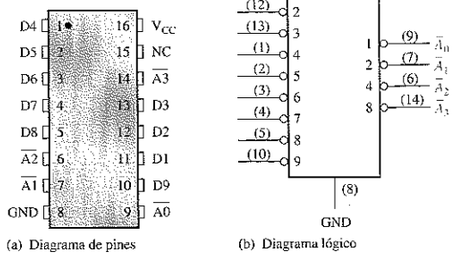


FIGURA 6.35
Diagrama de pines y símbolo lógico del codificador con prioridad decimal-BCD 74HC147 (HPRI, Highest value input has priority, la entrada de mayor valor tiene prioridad).

EJEMPLO 6.11 Si tenemos niveles BAJOS en los pines 1, 4 y 13 del 74HC147 que se muestra en la Figura 6.35, indicar el estado de sus cuatro salidas. Todas las demás entradas están a nivel ALTO.

Solución. El pin 4 es el dígito decimal de orden más alto que tiene una entrada a nivel BAJO y representa el número decimal 7. Por tanto, los niveles de salida indican el código BCD para el decimal 7, donde $\overline{A}_0$ es el bit menos significativo (LSB) y $\overline{A}_3$ es el bit más significativo (MSB). La salida $\overline{A}_0$ es un nivel BAJO, $\overline{A}_1$ es BAJO, $\overline{A}_2$ es un nivel BAJO y $\overline{A}_3$ es un nivel ALTO.

Problema relacionado. ¿Cuáles son las salidas del 74HC147 si todas sus entradas están a nivel BAJO? ¿Y si todas están a nivel ALTO?

Un codificador MSI 8 líneas a 3 líneas (octal-binario)

El 74F148 es un codificador con prioridad que tiene ocho entradas activas a nivel BAJO y tres salidas binarias activas a nivel BAJO, como se muestra en la Figura 6.36. Este dispositivo se puede utilizar para convertir entradas octales (recuérdese que los dígitos octales son del 0 al 7) en código binario de 3 bits. Para activar este dispositivo, la entrada de activación (enable input, *EI*) tiene que estar activa a nivel BAJO. También tiene una *EO* (salida de activación, enable output) y una salida *GS* para permitir la ampliación. La salida *EO* está a nivel BAJO cuando la entrada *EI* está a nivel BAJO y ninguna de las entradas (de 0 a 7) se encuentra activada. *GS* está a nivel BAJO cuando *EI* está a nivel BAJO y cualquiera de las entradas se encuentra activada.

El 74F148 puede ser ampliado a un codificador de 16 líneas a 4 líneas conectando la salida *EO* del codificador de mayor orden a la entrada *EI* del codificador de menor orden, y aplicando la operación negativa-OR a las correspondientes salidas binarias, como se muestra en la Figura 6.37. La salida *EO* se utiliza como cuarto y más significativo bit. Esta configuración particular produce salidas activas a nivel ALTO para los números binarios de cuatro bits.

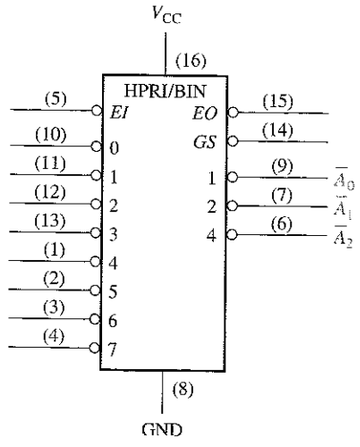


FIGURA 6.36
Símbolo lógico del codificador 8 líneas a 3 líneas 74F148.

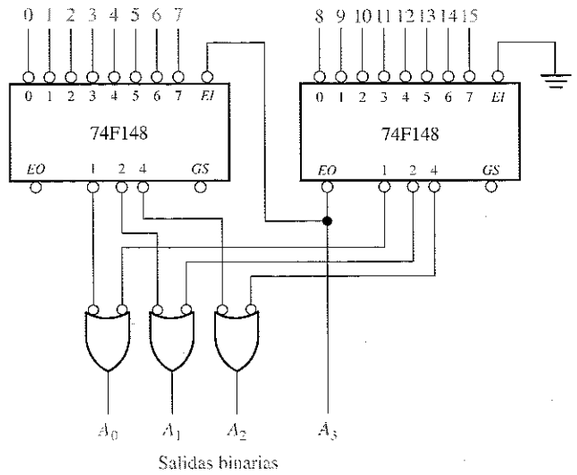


FIGURA 6.37
Codificador de 16 líneas a 4 líneas utilizando circuitos 74F148 y lógica externa.

Ejemplo de aplicación

El típico ejemplo de aplicación es un codificador de teclado. Por ejemplo, los diez dígitos decimales del teclado de una computadora tienen que codificarse para poder ser procesados por el circuito lógico. Cuando se pulsa una de las teclas, el dígito decimal se codifica a su correspondiente código BCD. La Figura 6.38 muestra la disposición de un sencillo codificador de teclado que utiliza un codificador con prioridad 74HC147. Las teclas se representan mediante diez pulsadores, conectados cada uno de ellos a una **resistencia de pull-up** (resistencia de conexión a la alimentación $+V_{CC}$). Las resistencias de pull-up aseguran que la línea esté a nivel ALTO cuando no haya ninguna tecla pulsada. Cuando se pulsa una tecla, la línea se conecta a tierra y se aplica un nivel BAJO a la correspondiente entrada del codificador. La tecla cero no está conectada, ya que la salida BCD es cero cuando ninguna de las otras teclas está pulsada.

La salida complementada BCD del codificador se conecta a un dispositivo de almacenamiento, de forma que los sucesivos códigos BCD se almacenan hasta que se haya introducido el número completo. En los siguientes capítulos veremos los métodos de almacenamiento de números BCD y de datos binarios.

REVISIÓN DE LA SECCIÓN 6.5

1. Supongamos que, a las entradas 2 y 9 del circuito de la Figura 6.34 se les aplica un nivel ALTO.
   (a) ¿Cuáles son los estados de las líneas de salida?
   (b) ¿Representa esto un código BCD válido?
   (c) ¿Cuál es la restricción de la lógica del codificador de la Figura 6.34?
2. **(a)** ¿Cuál es la salida $\bar{A}_3\bar{A}_2\bar{A}_1\bar{A}_0$ cuando se aplican niveles BAJOS de tensión a los pines 1 y 5 del 74HC147 de la Figura 6.35?
   (b) ¿Qué representa esta salida?

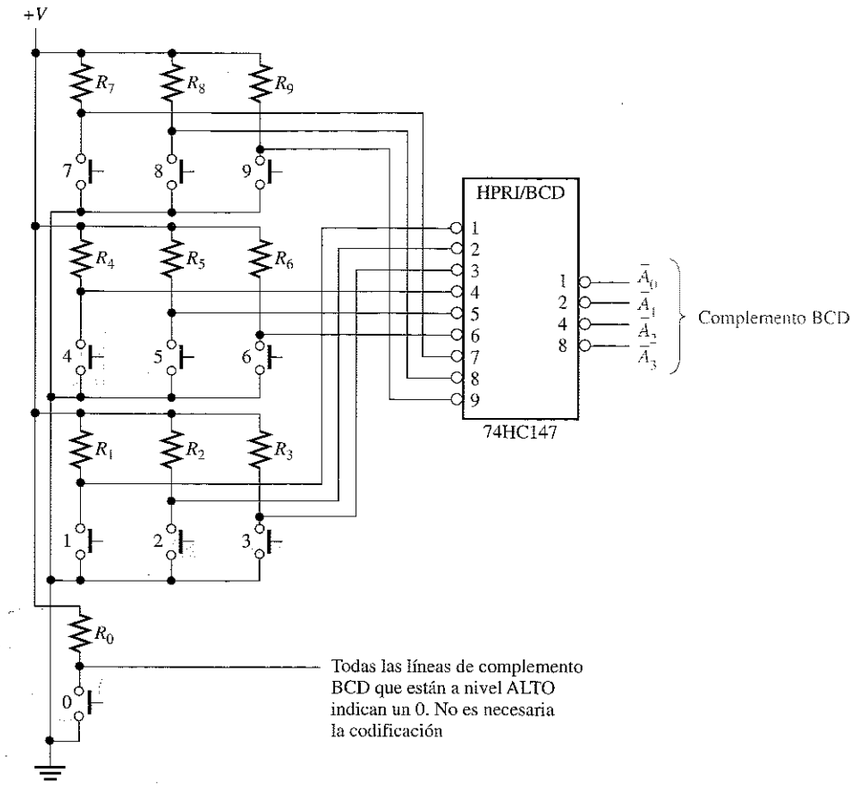


FIGURA 6.38
Codificador de un teclado simplificado.

6.6 ■ CONVERTIDORES DE CÓDIGO

En esta sección, vamos a examinar algunos métodos que utilizan circuitos lógicos combinacionales para pasar de un código a otro.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar el proceso de conversión de BCD a binario. ■ Utilizar el 74184 y el 74185 en la conversión BCD-binario y binario-BCD, respectivamente. ■ Ampliar el 74184 y el 74185 para poder manejar números binarios BCD mayores. ■ Utilizar puertas OR-exclusiva en la conversión entre código binario y código Gray.

Conversión BCD-binario

Uno de los métodos de conversión de código BCD a binario utiliza circuitos sumadores. El proceso básico de conversión consiste en lo siguiente:

1. El valor, o peso, de cada uno de los bits de un número BCD se representa por un número binario.
2. Se suman todas las representaciones binarias de los pesos de los bits del número BCD que son 1.
3. El resultado de la suma es el equivalente binario del número BCD.

Una manera más concisa de expresar esta operación es:

Para obtener el número binario completo hay que sumar los números binarios que representan los pesos de los bits del número BCD.

Vamos a examinar un código BCD de 8 bits (que representa un número decimal de 2 dígitos) para comprender la relación entre el código binario y el BCD. Por ejemplo, ya sabemos que el número decimal 87 se puede expresar en BCD como:

$$\underbrace{1000}_{8}\quad\underbrace{0111}_{7}$$

El grupo de 4 bits más a la izquierda representa 80 y el grupo de 4 bits más a la derecha representa el número 7. Esto quiere decir: el grupo más a la izquierda tiene un peso de 10 y el de más a la derecha tiene un peso de 1. Dentro de cada grupo, el peso binario de cada bit es el siguiente:

	Dígito de las decenas				*Dígito de las unidades*			
Peso:	80	40	20	10	8	4	2	1
Designación de bit:	B_3	B_2	B_1	B_0	A_3	A_2	A_1	A_0

El equivalente binario de cada bit BCD es un número binario que representa el peso de cada bit dentro del número BCD completo. Esta representación se muestra en la Tabla 6.7.

TABLA 6.7 ***Representación binaria de los pesos de los bits BCD.***

Bit BCD	Peso BCD	(MSB) 64	32	16	8	4	2	(LSB) 1
A_0	1	0	0	0	0	0	0	1
A_1	2	0	0	0	0	0	1	0
A_2	4	0	0	0	0	1	0	0
A_3	8	0	0	0	1	0	0	0
B_0	10	0	0	0	1	0	1	0
B_1	20	0	0	1	0	1	0	0
B_2	40	0	1	0	1	0	0	0
B_3	80	1	0	1	0	0	0	0

Si las representaciones binarias de los pesos de todos los 1s del número BCD se suman, el resultado es el número binario que corresponde al número BCD. El Ejemplo 6.12 ilustra esto.

EJEMPLO 6.12 Convertir a binario los números BCD 00100111 (27 decimal) y 10011000 (98 decimal).

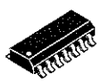
Solución. Escribir la representación binaria de los pesos de todos los 1s que aparecen en los números y, a continuación, sumarlos todos.

```
80 40 20 10 8 4 2 1
0  0  1  0  0 1 1 1
                  |  |  |
                  |  |  |  ---> 0000001   1
                  |  |  ---> 0000010   2
                  |  ---> 0000100   4
                  ---> +0010100   20
                          0011011   Número binario que representa el decimal 27

80 40 20 10 8 4 2 1
1  0  0  1  1 0 0 0
|        |  |
|        |  |  ---> 0001000   8
|        |  ---> 0001010   10
|  ---> +1010000   80
                  1100010   Número binario que representa el decimal 98
```

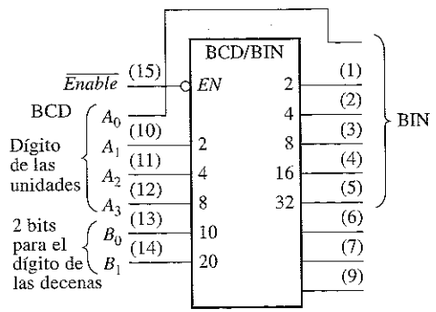
Problema relacionado. Explicar el proceso de conversión del número BCD 01000001 a binario.

Teniendo este procedimiento básico en mente, vamos a determinar cómo podemos implementar este proceso mediante circuitos lógicos. Una vez que se determina la representación binaria de cada 1 del número BCD, se pueden emplear circuitos sumadores para sumar los 1s de cada columna de la representación binaria. Los 1s aparecen en una determinada columna, sólo cuando los correspondientes bits BCD valen 1. Por tanto, la aparición de un 1 BCD puede utilizarse para generar el 1 binario apropiado en la columna correspondiente de la estructura del sumador. Para manejar un número decimal de dos cifras (dos potencias de diez), en código BCD, necesitamos 8 líneas BCD de entrada y siete líneas binarias de salida. Se necesitan 7 bits binarios para representar números de 0 a 99.

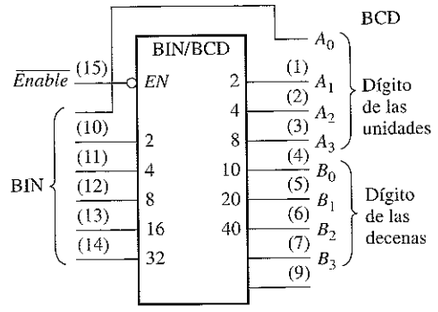


Convertidores MSI BCD-binario y binario-BCD

Los **convertidores de código MSI** se suelen implementar con dispositivos lógicos programables (PLD, programmable logic device), que se presentarán en el Capítulo 7. Ejemplos de estos dispositivos son el 74184, un dispositivo PLD preprogramado como convertidor BCD-binario, y el 74185, dispositivo PLD preprogramado como un convertidor binario-BCD. Los símbolos lógicos de estos dispositivos utilizados como convertidores de 6 bits se muestran en la Figura 6.39. Las Figuras 6.40 y 6.41 muestran cómo estos dispositivos pueden ampliarse para convertir números con mayor cantidad de bits.



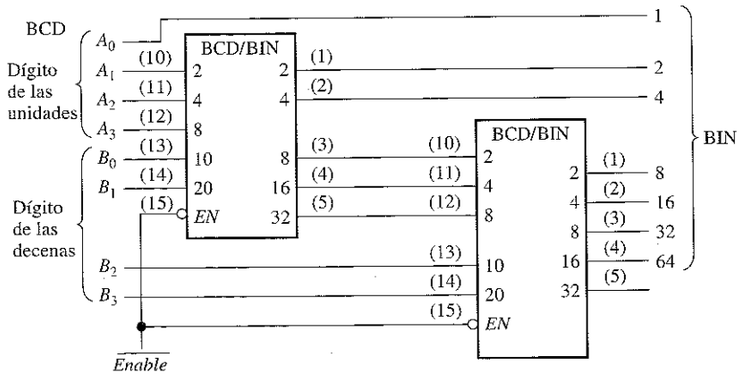
(a) Convertidor BCD-binario de 6 bits 74184



(b) Convertidor BCD-binario de 6 bits 74185

FIGURA 6.39

FIGURA 6.40
Dispositivos 74184 para la conversión de dos dígitos BCD a binario.



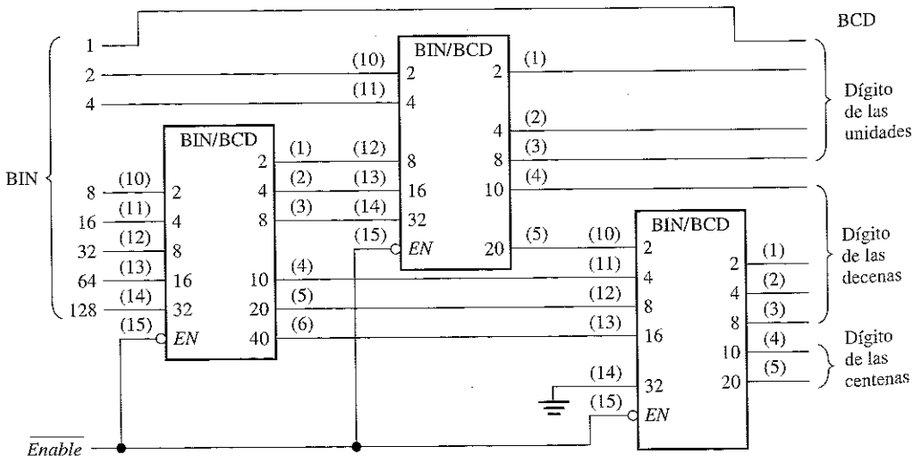


FIGURA 6.41
Dispositivos 74185 para la conversión de ocho bits binarios a BCD.

Conversión binario-Gray y Gray-binario

El proceso básico de conversión de código Gray a binario fue tratado en el Capítulo 2. Ahora vamos a ver cómo se pueden utilizar puertas OR-exclusiva en estas conversiones. Los dispositivos lógicos programables (PLD) también se pueden utilizar para realizar estas conversiones de código. La Figura 6.42 muestra un convertidor de 4 bits binarios a código Gray, y la Figura 6.43 ilustra un convertidor de 4 bits Gray a binario.

FIGURA 6.42
Lógica de conversión de 4 bits binarios a Gray.

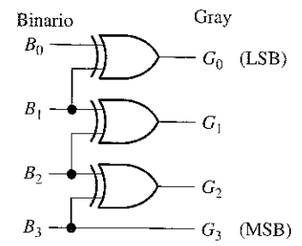
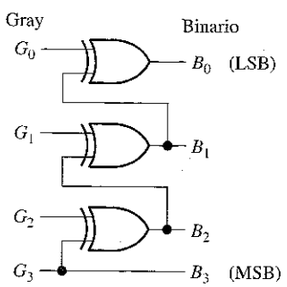


FIGURA 6.43
Lógica de conversión de 4 bits Gray a binario.



EJEMPLO 6.13 **(a)** Convertir el número binario 0101 a código Gray utilizando puertas OR-exclusiva.
(b) Convertir el código Gray 1011 a binario utilizando puertas OR-exclusiva.

Solución

(a) 0101_2 es 0111 en código Gray. Véase la Figura 6.44(a).
(b) 1011 en código Gray es 1101_2. Véase la Figura 6.44(b).

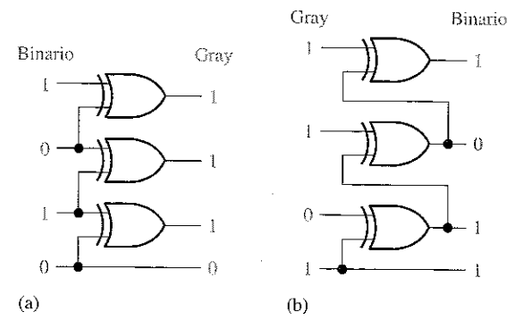


FIGURA 6.44

Problema relacionado. ¿Cuántas puertas OR-exclusiva son necesarias para convertir 8 bits binarios a código Gray?

REVISIÓN DE LA SECCIÓN 6.6

1. Convertir el número BCD 10000101 a binario.
2. Dibujar el diagrama lógico para convertir a código Gray un número binario de 8 bits.

6.7 ■ MULTIPLEXORES (SELECTORES DE DATOS)

Un multiplexor (MUX) es un dispositivo que permite dirigir la información digital procedente de diversas fuentes a una única línea para ser transmitida a través de dicha línea a un destino común. El multiplexor básico posee varias líneas de entrada de datos y una única línea de salida. También posee entradas de selección de datos, que permiten conmutar los datos digitales provenientes de cualquier entrada hacia la línea de salida. A los multiplexores también se les conoce como selectores de datos.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar el funcionamiento básico de un multiplexor. ■ Describir los multiplexores MSI 74HC151 y 74HC157A. ■ Ampliar un multiplexor para poder manejar mayor cantidad de entradas de datos. ■ Utilizar un multiplexor como generador de funciones lógicas.

El símbolo lógico de un **multiplexor (MUX)** de cuatro entradas se muestra en la Figura 6.45. Obsérvese que dispone de dos líneas de selección de datos, dado que con dos bits se puede seleccionar cualquiera de las cuatro líneas de entrada de datos.

En la Figura 6.45, un código binario de dos bits en las entradas de selección de datos (S) va a permitir que los datos de la entrada seleccionada pasen a la salida de datos. Si aplicamos un 0 binario ($S_1 = 0$ y $S_0 = 0$) a las líneas de selección de datos, los datos de la entrada D_0 aparecerán en la línea de datos de salida. Si aplicamos un 1 binario ($S_1 = 0$ y $S_0 = 1$), los datos de la entrada D_1 aparecerán en la salida de datos. Si se aplica un 2 binario ($S_1 = 1$ y $S_0 = 0$), obtendremos en la salida los datos de D_2. Si aplicamos un 3 binario ($S_1 = 1$ y $S_0 = 1$), los datos de D_3 serán conmutados a la línea de salida. El resumen del funcionamiento se puede ver en la Tabla 6.8.

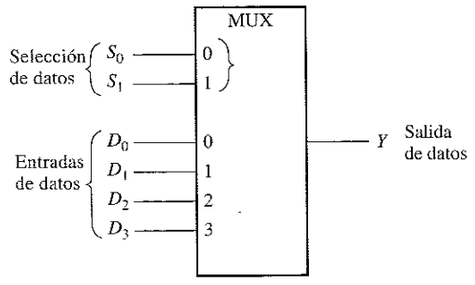


FIGURA 6.45
Símbolo lógico de un selector/multiplexor de datos de una salida y cuatro entradas.

TABLA 6.8
Selección de datos de un multiplexor de 1 salida y 4 entradas.

Entradas de selección de datos		Entrada seleccionada
S_1	S_0	
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

NOTAS INFORMÁTICAS
Un *bus* es una ruta interna por la que se envían señales eléctricas desde una parte a otra de una computadora. En las redes de computadoras, un *bus compartido* es aquel que está conectado a todos los microprocesadores del sistema, con el fin de intercambiar datos. Un bus compartido puede contener dispositivos de memoria y de entrada/salida a los que pueden acceder todos los microprocesadores del sistema. El acceso al bus compartido es controlado mediante un *árbitro de bus* (una especie de multiplexor), el cual hace que sólo un procesador utilice cada vez el bus compartido del sistema.

Vamos a fijarnos ahora en la circuitería lógica necesaria para realizar la multiplexación. La salida de datos es igual a la entrada de datos *seleccionada*. Podemos, por consiguiente, derivar una expresión lógica para la salida en función de la entrada de datos y de las entradas de selección de datos.

La salida de datos es igual a D_0 si y sólo si $S_1 = 0$ y $S_0 = 0$: $Y = D_0\overline{S}_1\overline{S}_0$

La salida de datos es igual a D_1 si y sólo si $S_1 = 0$ y $S_0 = 1$: $Y = D_1\overline{S}_1S_0$

La salida de datos es igual a D_2 si y sólo si $S_1 = 1$ y $S_0 = 0$: $Y = D_2S_1\overline{S}_0$

La salida de datos es igual a D_3 si y sólo si $S_1 = 1$ y $S_0 = 1$: $Y = D_3S_1S_0$

Cuando sumamos lógicamente (operación OR) estos términos, la expresión total para los datos de salida será:

$$Y = D_0\overline{S}_1\overline{S}_0 + D_1\overline{S}_1S_0 + D_2S_1\overline{S}_0 + D_3S_1S_0$$

La implementación de esta ecuación requiere cuatro puertas AND de tres entradas, una puerta OR de cuatro entradas y dos inversores para generar los complementos de S_1 y S_0, como se muestra en la Figura 6.46. Dado que los datos pueden ser seleccionados desde cualquier línea de entrada, se conoce también a este circuito con el nombre de **selector de datos**.

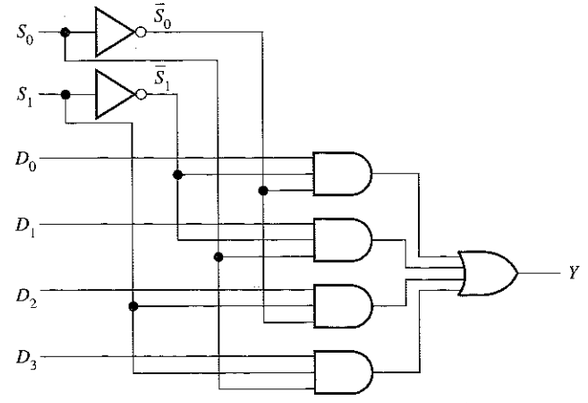
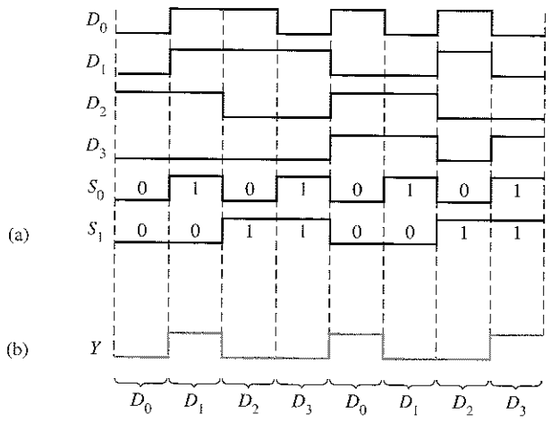


FIGURA 6.46
Diagrama lógico de un multiplexor de cuatro entradas.

EJEMPLO 6.14 Se aplican las formas de onda de la Figura 6.47(a) a la entrada de datos y a la entrada de selección del multiplexor de la Figura 6.46. Determinar la señal de salida en relación a las entradas.

FIGURA 6.47



Solución. El estado binario de las entradas de selección de datos durante cada intervalo determina cuál es la entrada de datos seleccionada. Obsérvese que las entradas de selección de datos siguen la secuencia binaria repetitiva 00, 01, 10, 11, 00, 01, 10, 11, etc. La forma de onda de salida resultante se muestra en la Figura 6.47(b).

Problema relacionado. Construir un diagrama de tiempos que muestre todas las entradas y la salida, si las formas de onda de S_0 y S_1 de la Figura 6.47 se intercambian.

Cuádruple multiplexor/selector de datos de 2 entradas MSI

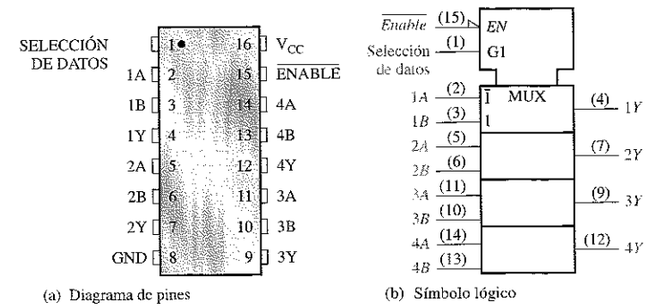
El 74HC157A, al igual que su versión LS, está formado por cuatro multiplexores de dos entradas. Cada uno de los cuatro multiplexores comparten una misma línea de selección de datos y una de *habilitación* (*enable*). Ya que sólo existen dos entradas de datos que puedan ser seleccionadas en cada multiplexor, es suficiente con tener una única entrada de selección.

Un nivel BAJO en la entrada de *habilitación* ($\overline{Enable}$) permite al dato de entrada seleccionado pasar a la salida. Un nivel ALTO en la entrada $\overline{Enable}$ evita que los datos pasen a la salida, es decir, inhabilita los multiplexores.

Símbolos lógicos ANSI/IEEE. En la Figura 6.48(a) se muestra el diagrama de pines del 74HC157A y su símbolo lógico ANSI/IEEE en la Figura 6.48(b). Podemos observar que los cuatro multiplexores se representan mediante divisiones del bloque y que las entradas comunes a los cuatro multiplexores se indican como entradas al bloque recortado de la parte superior, que recibe el nombre de *bloque común de control*. Todas las etiquetas dentro del bloque superior del MUX se aplican a los bloques que haya por debajo.

Obsérvense las etiquetas 1 y $\overline{1}$ de los bloques del MUX y la etiqueta G1 en el bloque común de control. Estas etiquetas son un ejemplo del sistema de **notación de dependencia** especificado en el estándar ANSI/IEEE 91-1984. En este caso, G1 indica una relación AND entre la entrada de selección de datos y las entradas de datos designadas por 1 o $\overline{1}$. El $\overline{1}$ indica que la relación AND se aplica al complemento de la entrada G1. En otras palabras, cuando la entrada de selección está a nivel ALTO, se seleccionan las entradas *B* de los multiplexores y, cuando la entrada de selección está a nivel BAJO, se seleccionan las entradas *A*. Para indicar dependencia AND siempre se usa una "G". Otros aspectos de la notación de dependencia serán tratados a lo largo del libro.

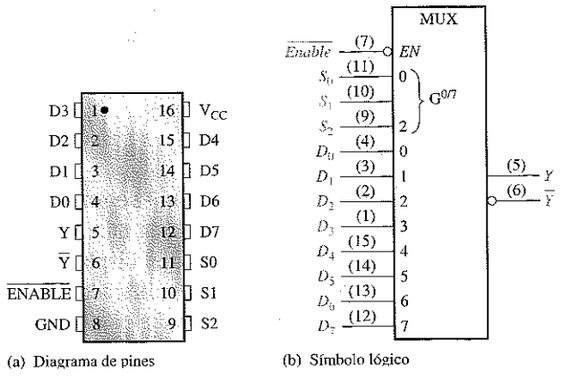
FIGURA 6.48
Diagrama de pines y símbolo lógico para el cuádruple selector de datos/multiplexor de dos entradas 74HC157A.



Multiplexor/selector de datos MSI de 8 entradas

El 74HC151 tiene ocho entradas de datos ($D_0 - D_7$) y, por tanto, tres líneas de entrada de selección de datos ($S_0 - S_2$). Se necesitan tres bits para seleccionar cualquiera de las ocho entradas de datos ($2^3 = 8$). Un nivel BAJO en la entrada de habilitación ($\overline{Enable}$) permite que los datos de entrada seleccionados pasen a la salida. Obsérvese que se encuentran disponibles tanto la salida de datos como su complemento. En la Figura 6.49(a) se muestra el diagrama de pines y en la parte (b) el símbolo lógico ANSI/IEEE. En este caso no hay necesidad de tener un bloque de control común en el símbolo lógico, ya que sólo hay que controlar un único multiplexor, y no cuatro como en el 74HC157A. La etiqueta G $^{0/7}$ dentro del símbolo lógico indica la relación AND entre las entradas de selección de datos y cada una de las entradas de datos, de la 0 a la 7.

FIGURA 6.49
Diagrama de pines y símbolo lógico para el multiplexor/selector de datos de 8 entradas 74HC151.



EJEMPLO 6.15 Utilizar multiplexores 74HC151 y cualquier otra lógica necesaria para multiplexar 16 líneas de datos en una única línea de salida de datos.

Solución. En la Figura 6.50 se muestra una implementación de este sistema. Se necesitan cuatro bits para seleccionar cualquiera de las 16 líneas de entrada de datos ($2^4 = 16$). En esta aplicación, la entrada de habilitación ($\overline{Enable}$) se utiliza como el bit más significativo de selección de datos. Cuando el MSB del código de selección de datos está a nivel BAJO, se habilita el 74HC151 de la izquierda y se selecciona una de las entradas de datos (D_0 a la D_7) mediante los otros tres bits de selección de datos. Cuando el MSB de selección de datos está a nivel ALTO, se habilita el 74HC151 de la derecha y se selecciona una de las entradas de datos (D_8 a la D_{15}). Los datos de entrada seleccionados pasan luego a través de la puerta negativa-OR y van a dar a la única línea de salida.

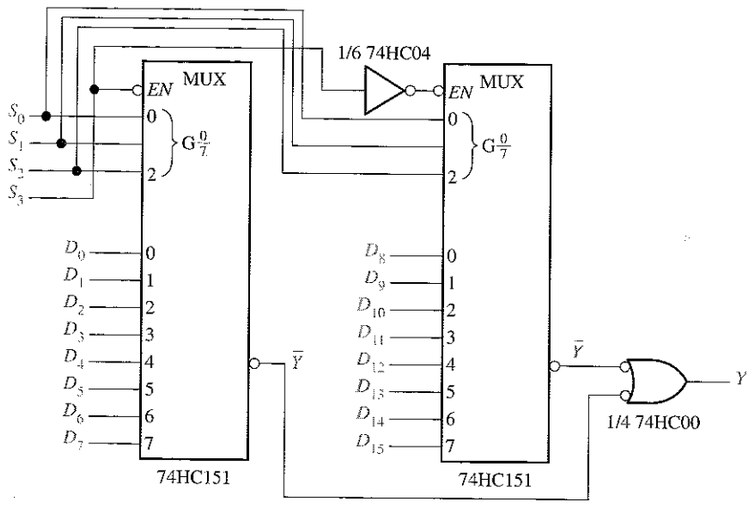


FIGURA 6.50
Multiplexor de 16 entradas.

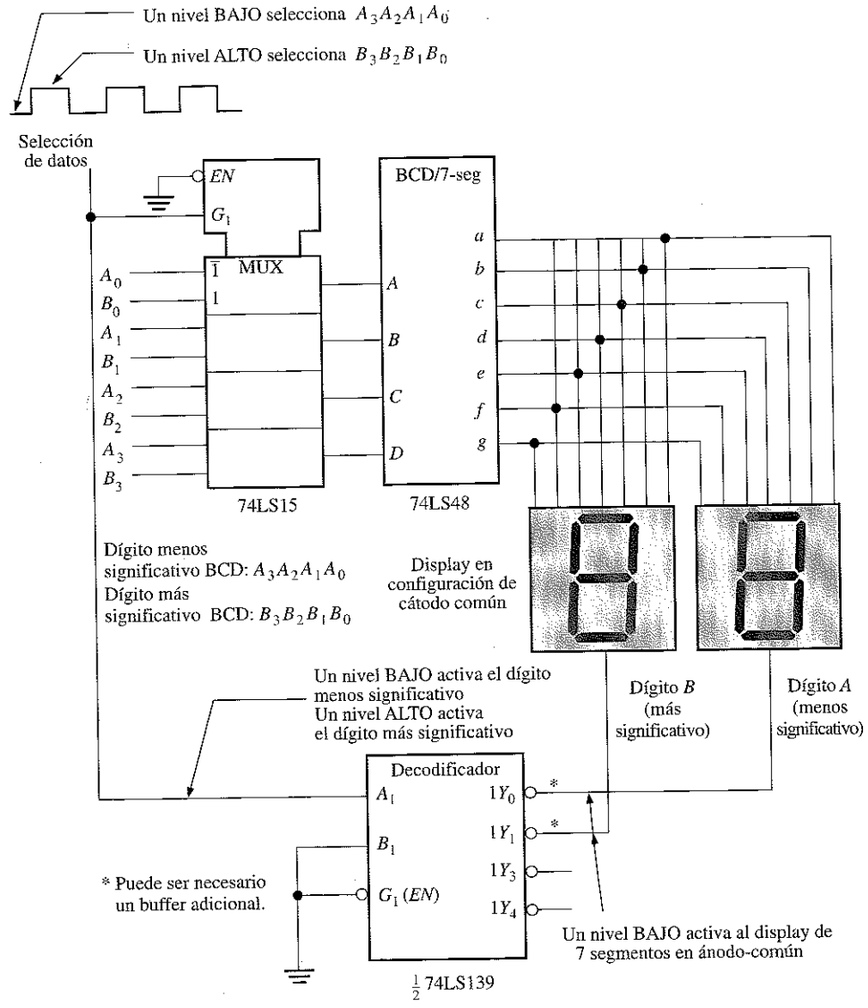
Problema relacionado. Determinar los códigos necesarios en las entradas de selección de datos para seleccionar cada una de las siguientes entradas de datos: D_0, D_4, D_8 y D_{13}.

Ejemplos de aplicación

Display multiplexor de 7 segmentos. La Figura 6.51 muestra un método simplificado de multiplexación de números BCD para un display de 7 segmentos. En este ejemplo, se visualizan en el display de 7 segmentos números de dos dígitos, mediante el uso de un único decodificador BCD a 7 segmentos. Este método básico de multiplexación puede ampliarse para visualizar números con cualquier cantidad de dígitos. Su funcionamiento básico es el siguiente.

Se aplican dos dígitos BCD ($A_3A_2A_1A_0$ y $B_3B_2B_1B_0$) a las entradas de un multiplexor. Se aplica una señal cuadrada a la línea de selección de datos de forma que, cuando está a nivel BAJO, los bits de A ($A_3A_2A_1A_0$) pasan a las entradas del decodificador BCD a 7 segmentos 74LS48. El nivel BAJO en la entrada de selección de datos genera un nivel BAJO en la entrada 1 del decodificador de 2 líneas a 4 líneas 74LS139, activando su salida 0 y habilitando el display del dígito A, al conectar su terminal común a masa. El dígito A se encuentra ahora encendido, mientras que el B está apagado.

FIGURA 6.51
Lógica de multiplexación simplificada de un display de 7 segmentos.



Cuando la línea de selección de datos pasa a nivel ALTO, los bits de B ($B_3B_2B_1B_0$) pasan a las entradas del decodificador BCD a 7 segmentos. Ahora se acti-

va la salida 1 del decodificador 74LS139, encendiendo el display del dígito *B*, que pasa a visualizarse, mientras que el *A* se encuentra apagado. El ciclo se repite a la frecuencia de la señal cuadrada que se aplica a la entrada de selección de datos. Esta frecuencia tiene que ser lo suficientemente alta (unos 30 Hz) para evitar el parpadeo en los displays cuando se multiplexa la presentación de los dígitos.

Generador de funciones lógicas. Una aplicación muy útil de los multiplexores/selectores de datos consiste en la generación de funciones lógicas combinacionales en forma de suma de productos. Cuando se emplea de esta manera, este dispositivo puede reemplazar puertas lógicas discretas, puede reducir significativamente el número de circuitos integrados y permite que los cambios en el diseño sean mucho más sencillos.

Con el fin de ilustrar esto, se ha utilizado un multiplexor/selector de datos de 8 entradas 74HC151, para implementar cualquier función lógica de 3 variables, conectando las variables a las entradas de selección y asignando a cada entrada de datos el nivel lógico requerido por la tabla de verdad para dicha función. Por ejemplo, si la función es 1 cuando la combinación de variables es $\overline{A}_2A_1\overline{A}_0$, la entrada 2 (seleccionada por 010) se conecta a un nivel ALTO. Este nivel ALTO pasa a la salida cuando esta combinación particular de variables ocurre en las líneas de selección de datos. Un ejemplo nos servirá para clarificar esta aplicación.

EJEMPLO 6.16 Implementar la función lógica especificada en la Tabla 6.9, utilizando un multiplexor/selector de datos de 8 entradas 74HC151. Comparar este método con una implementación discreta con puertas lógicas.

TABLA 6.9

Entradas			Salida
A_2	A_1	A_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Solución. Obsérvese en la tabla de verdad que Y vale 1 para las siguientes combinaciones de variables de entrada: 001, 011, 101 y 110. Para el resto de las combinaciones, Y vale 0. Para poder implementar esta función mediante el selector de datos, la entrada de datos seleccionada por cada una de las combinaciones mencionadas anteriormente tiene que conectarse a un nivel ALTO (5 V). El resto de las entradas de datos debe conectarse a un nivel BAJO (tierra), como se muestra en la Figura 6.52.

La implementación de esta función mediante puertas lógicas requeriría cuatro puertas AND de 3 entradas, una puerta OR de 4 entradas y tres inversores, a menos que la expresión pudiera ser simplificada.

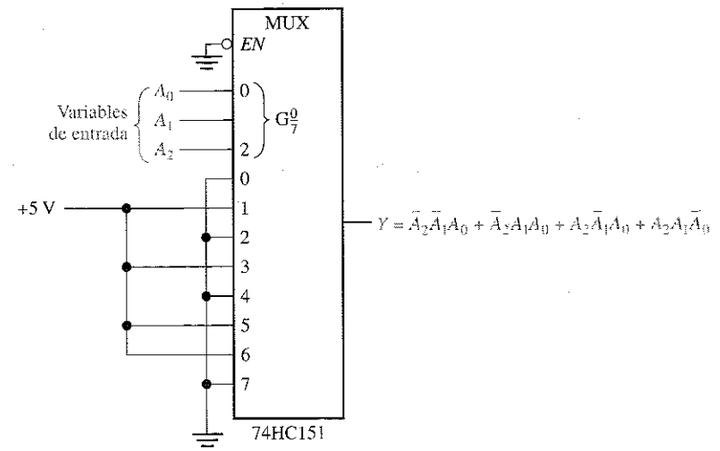


FIGURA 6.52
Multiplexor/selector de datos conectado como generador de funciones lógicas de 3 variables.

Problema relacionado. Utilizar un 74HC151 para implementar la siguiente expresión:

$$Y = \overline{A}_2\overline{A}_1\overline{A}_0 + A_2\overline{A}_1\overline{A}_0 + \overline{A}_2A_1\overline{A}_0$$

El Ejemplo 6.16 ilustra cómo se puede utilizar el selector de datos de 8 entradas como generador de funciones lógicas de tres variables. En realidad, este dispositivo puede usarse también como generador de funciones lógicas de 4 variables utilizando uno de los bits (A_0) junto con las entradas de datos.

Una tabla de verdad de 4 variables posee 16 combinaciones de las variables de entrada. Cuando se utiliza un selector de datos de 8 bits, cada entrada se selecciona dos veces: la primera vez cuando A_0 es 0 y la segunda cuando A_0 es 1. Teniendo esto en mente, se pueden aplicar las siguientes reglas (Y es la salida y A_0 es el bit menos significativo):

1. Si $Y = 0$ las dos veces que una entrada de datos dada se selecciona mediante una determinada combinación de las variables de entrada $A_3A_2A_1$, dicha entrada de datos se conecta a tierra (0).
2. Si $Y = 1$ las dos veces que una entrada de datos dada se selecciona mediante una determinada combinación de las variables de entrada $A_3A_2A_1$, dicha entrada de datos se conecta a $+V$ (1).
3. Si Y es diferente las dos veces que se selecciona una entrada por una determinada combinación de las variables de entrada $A_3A_2A_1$ y si $Y = A_0$, la entrada de datos se conecta a A_0.
4. Si Y es diferente las dos veces que se selecciona una entrada por una determinada combinación de las variables de entrada $A_3A_2A_1$ y si $Y = \overline{A}_0$, la entrada de datos se conecta a $\overline{A}_0$.

Los siguientes ejemplos ilustran este método.

EJEMPLO 6.17 Implementar la función lógica de la Tabla 6.10 mediante la utilización del multiplexor/selector de datos de 8 entradas 74HC151. Comparar este método con una implementación realizada con puertas lógicas discretas.

TABLA 6.10

Dígito decimal	Entradas A_3	A_2	A_1	A_0	Salida Y
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	1

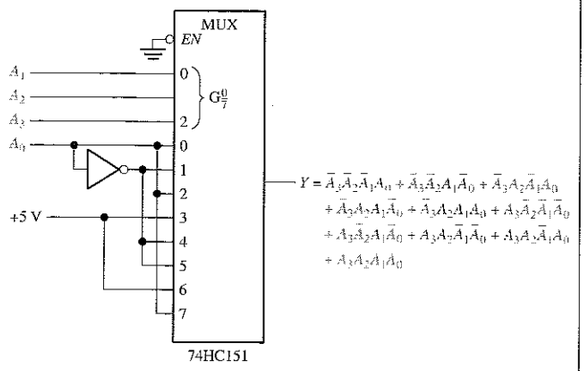


FIGURA 6.53
Multiplexor/selector de datos conectado como generador de funciones lógicas de 4 variables.

Solución.

Las entradas de selección de datos son $A_3A_2A_1$. En la primera fila de la tabla, $A_3A_2A_1 = 000$ e $Y = A_0$. En la segunda fila, de nuevo $A_3A_2A_1$ es 000, $Y = A_0$. Por tanto, A_0 se conecta a la entrada 0. En la tercera fila de la tabla $A_3A_2A_1 = 001$ e $Y = \bar{A}_0$. También, en la cuarta columna, $A_3A_2A_1$ es de nuevo 001 e $Y = \bar{A}_0$. Por tanto, A_0 se invierte y se conecta a la entrada 1. Este análisis continúa hasta que cada entrada se conecta apropiadamente de acuerdo con las reglas especificadas. La implementación se muestra en la Figura 6.53.

Si se implementara con puertas lógicas, la función requeriría al menos diez puertas AND de 4 entradas, una puerta OR de 10 entradas y cuatro inversores, aunque una posible simplificación reduciría este requisito.

Problema relacionado. Si, en la Tabla 6.10, $Y = 0$ cuando las entradas son todas cero y, alternativamente, 1 y 0 para las restantes filas de la tabla, implementar la función lógica resultante utilizando un 74HC151.

REVISIÓN DE LA SECCIÓN 6.7

1. En la Figura 6.46, $D_0 = 1$, $D_1 = 0$, $D_2 = 1$, $D_3 = 0$, $S_0 = 1$ y $S_1 = 0$. ¿Cuál es la salida?
2. Identificar cada dispositivo MSI:
   (a) 74LS157 **(b)** 74LS151
3. En las entradas de datos de un 74HC151 se aplican alternativamente niveles BAJOS y ALTOS, comenzando por $D_0 = 0$. Las líneas de selección de datos se secuencian mediante un contador binario (000, 001, 010, etc.) a una frecuencia de 1 kHz. La entrada de habilitación está a nivel BAJO. Describir la forma de onda de salida.
4. Describir brevemente el propósito de cada uno de los siguientes dispositivos de la Figura 6.51.
   (a) 74HC157A **(b)** 74LS48 **(c)** 74LS139

6.8 ■ DEMULTIPLEXORES

Un demultiplexor (DEMUX) básicamente realiza la función contraria a la del multiplexor. Toma datos de una línea y los distribuye a un determinado número de líneas de salida. Por este motivo, el demultiplexor se conoce también como distribuidor de datos. Como veremos, los decodificadores pueden utilizarse también como demultiplexores.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar el funcionamiento básico de un demultiplexor. ■ Describir cómo el decodificador de 4 líneas a 16 líneas 74HC154 puede utilizarse como demultiplexor. ■ Desarrollar el diagrama de tiempos de un demultiplexor con un número determinado de líneas de entrada y de líneas de selección de datos.

La Figura 6.54 muestra un circuito **demultiplexor (DEMUX)** de 1 línea a 4 líneas. La línea de entrada de datos está conectada a todas las puertas AND. Las dos líneas de selección de datos activan únicamente una puerta cada vez y los datos que aparecen en la línea de entrada de datos pasarán a través de la puerta seleccionada hasta la línea de salida de datos asociada.

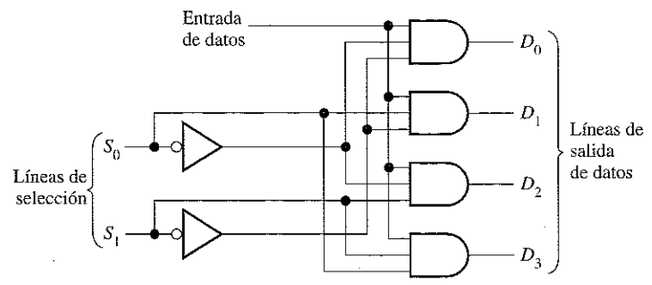


FIGURA 6.54
Demultiplexor de 1 línea a 4 líneas.

EJEMPLO 6.18 En la Figura 6.55 se muestra una forma de onda de entrada de datos serie y las entradas de selección de datos (S_0 y S_1). Determinar las formas de onda de datos de salida que obtendríamos en las salidas D_0 hasta la D_3 para el demultiplexor de la Figura 6.54.

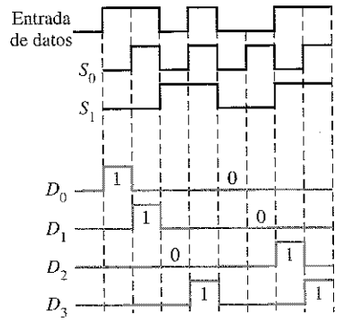


FIGURA 6.55

Solución. Las líneas de selección de datos reciben una secuencia binaria que hace que cada bit de entrada sucesivo sea redirigido hacia D_0, D_1, D_2 y D_3 secuencialmente, como se puede ver en la señal de salida de la Figura 6.55.

Problema relacionado. Desarrollar el diagrama de tiempos del demultiplexor si se invierten las señales S_0 y S_1.

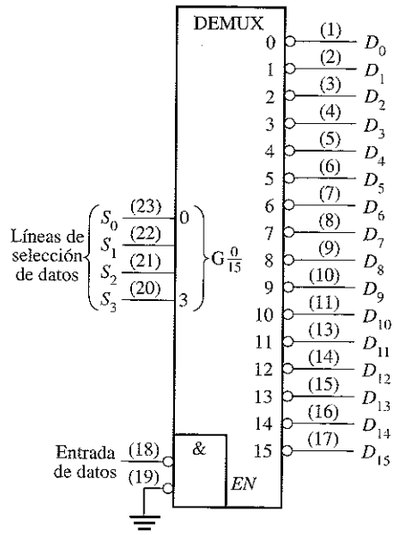
Un demultiplexor MSI

Hasta ahora hemos visto el 74HC154 como decodificador de 4 líneas a 16 líneas (Sección 6.4). Este dispositivo, así como otros decodificadores, se utiliza también en diversas aplicaciones como demultiplexor. El símbolo lógico de este circuito, cuando se emplea como demultiplexor, se muestra en la Figura 6.56. Cuando se utiliza con este fin, se usan las líneas de entrada como líneas de selección de datos; una de las entradas de activación del chip se usa como línea de entrada de datos y la otra se mantiene a nivel BAJO, para activar la puerta interna negativa-AND que se encuentra en la parte inferior del diagrama.

REVISIÓN DE LA SECCIÓN 6.8

1. En general, ¿cómo puede utilizarse un decodificador MSI como demultiplexor?
2. El demultiplexor 74HC154 de la Figura 6.56 tiene en las líneas de selección de datos el código binario 1010 y la línea de entrada de datos está a nivel BAJO. ¿Cuáles son los estados de las líneas de salida?

FIGURA 6.56
El 74HC154 utilizado como demultiplexor.



6.9 ■ GENERADORES / COMPROBADORES DE PARIDAD

Cuando se transfieren datos digitales de un punto a otro dentro de un sistema digital o cuando se transmiten códigos desde un sistema a otro, se pueden producir errores. Estos errores se manifiestan mediante cambios indeseados en los bits que conforman la información codificada; es decir, un 1 puede cambiar a 0 o un 0 a 1, debido a un mal funcionamiento de los componentes o al ruido eléctrico. En la mayoría de los sistemas digitales, la probabilidad de que haya un bit erróneo es muy pequeña, y la de que haya más de uno es todavía menor. En cualquier caso, cuando no se detecta un error, pueden originarse serios problemas en un sistema digital.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar el concepto de paridad. ■ Implementar un circuito de paridad básico con puertas OR-exclusiva. ■ Describir el funcionamiento de la lógica de un generador/comprobador de paridad básico. ■ Explicar el generador/comprobador de paridad de 9 bits 74LS280. ■ Explicar cómo se puede implementar en la transmisión de datos la detección de errores.

En el Capítulo 2 se vio el método de paridad de detección de errores, en el que se añade un **bit de paridad** a un grupo de bits de información para conseguir que el número total de 1s sea par o impar (dependiendo del sistema que se trate). Además de los bits de paridad, hay otros códigos específicos que también permiten realizar la detección de errores.

Lógica básica de la paridad

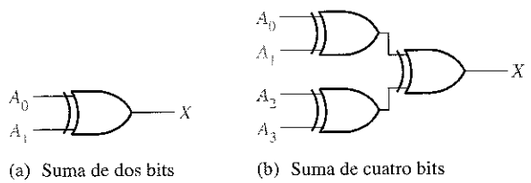
Para poder comprobar o generar la paridad adecuada dentro de un determinado código, se puede aplicar un principio muy sencillo:

La suma (descartando los acarreos) de un número par de 1s siempre es 0 y la suma de un número impar de 1s siempre es 1.

Por tanto, para determinar si un cierto código tiene **paridad par** o **paridad impar**, se suman todos los bits de ese código. Como sabemos, la suma de dos bits se puede generar mediante una puerta OR-exclusiva, como se muestra en la Figura 6.57(a); la suma de cuatro bits se puede realizar a partir de tres puertas OR-exclusiva conectadas como se indica en la Figura 6.57(b), y así sucesivamente. Cuando el número de 1s en las entradas es par, la salida X es 0 (nivel BAJO). Cuando el número de 1s es impar, la salida X es 1 (nivel ALTO).

NOTAS INFORMÁTICAS El microprocesador Pentium realiza comprobaciones internas de paridad, así como comprobaciones de paridad de los buses externos de direcciones y datos. En una operación de lectura, el sistema externo puede transferir la información de paridad junto con los bytes de datos. El microprocesador Pentium comprueba si la paridad resultante es par y envía la señal correspondiente. Cuando se envía un código de dirección, este microprocesador no lleva a cabo una comprobación de paridad de la dirección, pero sí que genera un bit de paridad para la dirección.

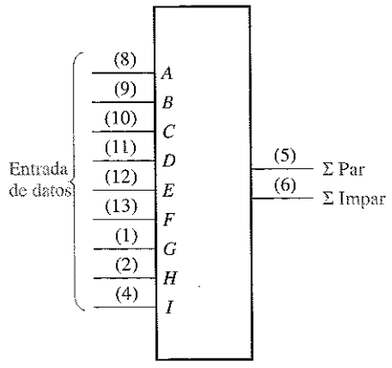
FIGURA 6.57



(a) Suma de dos bits (b) Suma de cuatro bits

Un generador/comprobador de paridad MSI de 9 bits

El símbolo lógico y la tabla de funciones de un 74LS280 se representa en la Figura 6.58. Este dispositivo MSI se puede utilizar para comprobar la paridad par o impar en un código de 9 bits (ocho bits de datos y un bit de paridad), o puede también emplearse para generar un bit de paridad para un código binario de hasta 9 bits. Sus entradas son desde A hasta I; cuando en las entradas hay un número par de 1s, la salida Σ Par es un nivel ALTO y la salida Σ Impar es un nivel BAJO.



(a) Símbolo lógico tradicional

Número de entradas A hasta I que están a nivel ALTO	Salidas	
	Σ Par	Σ Impar
0, 2, 4, 6, 8	H	L
1, 3, 5, 7, 9	L	H

(b) Tabla de funciones

FIGURA 6.58
El generador/comprobador de paridad de 9 bits 74LS280

Comprobador de paridad. Cuando este dispositivo se utiliza como un comprobador de paridad par, el número de bits de entrada deberá ser siempre par; y cuando se produzca un error, la salida Σ Par pasará a nivel BAJO y la salida Σ Impar será un nivel ALTO. Cuando se emplea como comprobador de paridad impar, el número de bits de entrada deberá ser siempre impar, y cuando se produzca un error, la salida Σ Impar será un nivel BAJO y la salida Σ Par será un nivel ALTO.

Generador de paridad. Si este dispositivo se utiliza como generador de paridad par, el bit de paridad se toma en la salida Σ Impar, ya que esta salida es 0 cuando hay un número par de bits de entrada y 1 cuando hay un número impar. Cuando se emplea como generador de paridad impar, el bit de paridad se toma en la salida Σ Par, dado que ésta es 0 cuando el número de bits de entrada es impar.

Sistema de transmisión de datos con detección de errores

En la Figura 6.59 se muestra un sistema simplificado de transmisión de datos para ilustrar una aplicación de los generadores/comprobadores de paridad, así como de los multiplexores y demultiplexores, y para ilustrar la necesidad de dispositivos de almacenamiento de datos en algunas aplicaciones.

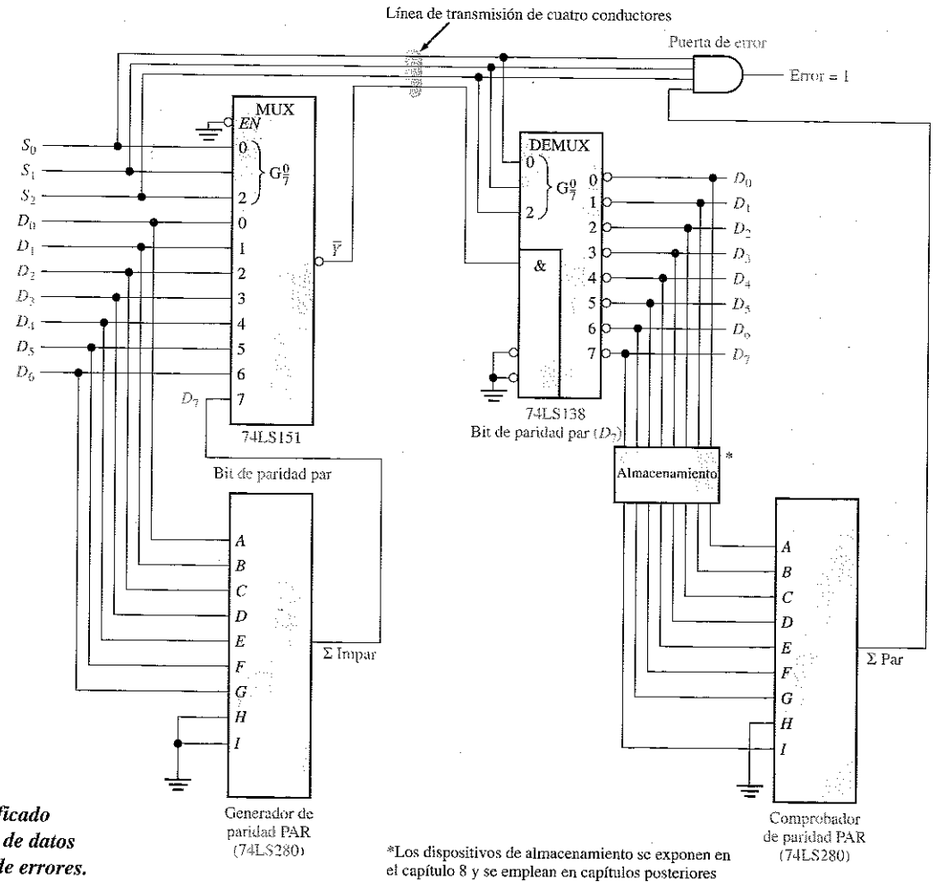


FIGURA 6.59
Sistema simplificado de transmisión de datos con detección de errores.

*Los dispositivos de almacenamiento se exponen en el capítulo 8 y se emplean en capítulos posteriores

En esta aplicación, los datos digitales procedentes de siete fuentes se multiplexan en una única línea para ser transmitidos a un punto distante. Se aplican los siete bits de datos (D_0 hasta D_6) a las entradas de datos del multiplexor y, al mismo tiempo, a las entradas del generador de paridad par. La salida Σ Impar del generador de paridad se utiliza como bit de paridad par. Este bit es 0 si el número de 1s en las entradas de la A a la I es par, y es 1 si el número de 1s en las mismas entradas es impar. Éste es el bit D_7 del código transmitido.

Las entradas de selección de datos van pasando cíclicamente por una secuencia binaria y cada bit de datos, comenzando en D_0, se transmite en serie por la línea de transmisión ($\overline{Y}$). En este ejemplo, la línea de transmisión está formada por cuatro conductores: uno para los datos serie y los otros tres para las señales de temporización (selección de datos). Existen maneras más sofisticadas de enviar información de temporización, pero estamos usando este método directo para ilustrar un principio básico.

En el extremo demultiplexor del sistema, las señales de selección de datos y la cadena de datos serie se aplican al demultiplexor. Los bits de datos se distribuyen mediante el demultiplexor a las líneas de salida en el orden en que llegaron a las entradas del multiplexor. Es decir, D_0 llega a la salida D_0, D_1 llega a la salida D_1, etc. El bit de paridad llega a la salida D_7. Estos ocho bits se almacenan temporalmente y se aplican al comprobador de paridad par. No todos estos bits se encuentran presentes en las entradas del comprobador de paridad hasta que el bit de paridad D_7 aparece y se almacena. En este instante, la puerta de error es activada por el código de selección de datos 111. Si la paridad es correcta, aparece un 0 en la salida Σ Par, manteniendo la salida ERROR a nivel 0. Si la paridad es incorrecta, todos los 1s aparecerán en las entradas de la puerta de error, lo que da lugar a un 1 en la salida ERROR.

Esta aplicación particular demuestra que es necesario disponer de algún dispositivo de almacenamiento de datos, por lo que, cuando en el Capítulo 8 estudiemos los dispositivos de almacenamiento y los utilicemos en capítulos posteriores, seremos capaces de apreciar mejor su utilidad.

El diagrama de tiempos que se muestra en la Figura 6.60 ilustra el caso específico de transmisión de dos palabras de 8 bits, una de las cuales tiene paridad correcta y la otra se transmite con un error.

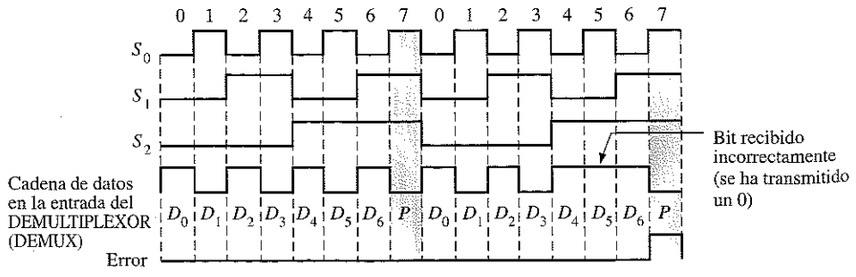


FIGURA 6.60
Ejemplo de transmisión de datos con y sin error en el sistema de la Figura 6.59.

REVISIÓN DE LA SECCIÓN 6.9

1. Añadir un bit de paridad par a cada uno de los siguientes códigos:
   (a) 110100 **(b)** 01100011
2. Añadir un bit de paridad impar a cada uno de los siguientes códigos:
   (a) 1010101 **(b)** 1000001
3. Comprobar cada uno de los siguientes códigos con paridad par e indicar si contienen errores.
   (a) 100010101 **(b)** 1110111001

6.10 ■ LOCALIZACIÓN DE AVERÍAS

En esta sección, se introduce y examina el problema de los impulsos de muy corta duración (glitches) en los decodificadores desde un punto de vista práctico. Un glitch es un pico de tensión o de corriente (impulso) no deseado de muy corta duración. Los circuitos lógicos pueden interpretar estos impulsos como una señal válida, originando fallos en el funcionamiento del circuito.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar qué es un glitch. ■ Determinar la causa de los glitches en una aplicación de un decodificador. ■ Utilizar el método de la validación de salida para eliminar glitches.

El circuito 74LS138 se ha utilizado como demultiplexor en el sistema de transmisión de datos de la Figura 6.59. Ahora, en la Figura 6.61, se utiliza como un decodificador de 3 líneas a 8 líneas (binario-octal), para ilustrar cómo pueden ocurrir los **glitches** y cómo identificar su origen. Las entradas $A_2A_1A_0$ del decodificador se secuencian mediante un contador binario y las señales resultantes de entrada y salida se pueden visualizar en la pantalla de un analizador lógico, como se muestra en la Figura 6.61.

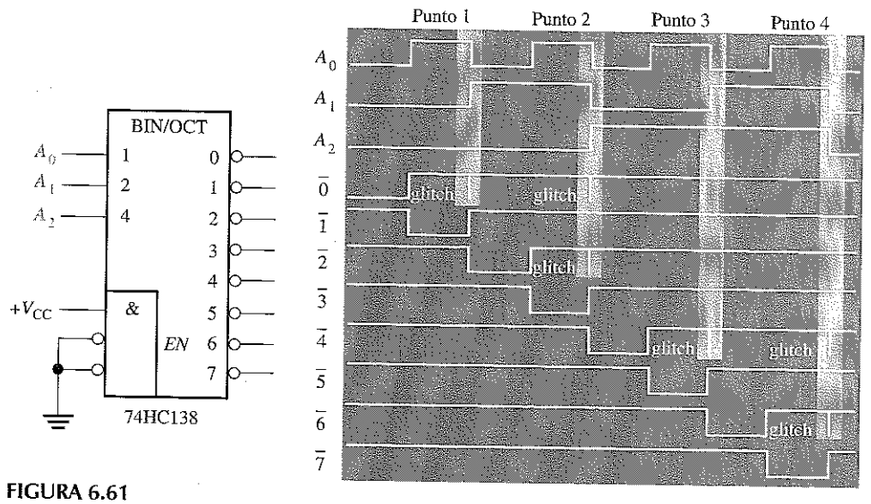


FIGURA 6.61
Señales del decodificador con glitches en la salida.

Las transiciones de la señal A_2 están retrasadas con respecto a las transiciones de A_1, y las de A_1 respecto a las transiciones de A_0. Esto es lo que suele ocurrir cuando se emplea un contador binario para generar las señales, como veremos en el Capítulo 9.

Las señales de salida son correctas excepto por los glitches que aparecen en algunas de ellas. Se puede utilizar un osciloscopio o un analizador lógico para visualizar los glitches, que normalmente son difíciles de ver. Generalmente, es preferible el analizador lógico, especialmente para detectar las velocidades de repetición bajas (menores de 10 kHz) y/o su ocurrencia irregular, ya que la mayoría de los analizadores lógicos dispone de la función de *captura de glitches*. Los osciloscopios se pueden emplear para observar los glitches con cierta seguridad, especialmente si éstos se producen con una velocidad de repetición alta y constante (mayor que 10 kHz).

Los puntos de interés, que son las zonas marcadas en las señales de entrada de la Figura 6.61, se visualizan como se muestra en la Figura 6.62. En el punto 1, se produce una transición por el estado 000 debido a las diferencias de los retardos de las señales. Esto origina el primer glitch en la salida $\overline{0}$ del decodificador. En el punto 2, aparecen dos estados de transición, 010 y 000. Éstos originan un glitch en la salida $\overline{2}$ del decodificador y un segundo glitch en la salida $\overline{0}$, respectivamente. En el punto 3, el estado de transición es 100, lo que origina el primer glitch en la salida $\overline{4}$ del decodificador. En el punto 4, los dos estados de transición son 110 y 100, los cuales originan el glitch en la salida $\overline{6}$ del decodificador y en la salida $\overline{4}$, respectivamente.

Una manera de eliminar este problema es aplicar impulsos de **validación** (strobing), que consiste en activar el decodificador mediante un impulso de validación (strobe) únicamente durante los intervalos de tiempo en que las señales no se encuentran en un estado de transición. Este método se ilustra en la Figura 6.63.

Consejos Prácticos

Además de los glitches que son el resultado de los retardos de propagación, como hemos visto en el caso de un decodificador, existen otros tipos de impulsos de ruido no deseados que pueden constituir también un problema. Los impulsos de corriente y tensión en las líneas de masa y alimentación (V_{CC}) son debidos a las señales de conmutación rápida en los circuitos digitales. Este problema se puede minimizar realizando un apropiado diseño de la placa de circuito impreso. Los impulsos de conmutación pueden ser absorbidos mediante el desacoplo de la tarjeta de circuito impreso con condensadores de 1 μF entre V_{CC} y masa. También deberían distribuirse condensadores más pequeños de desacoplo (0,022 μF a 0,1 μF) en distintos puntos de la placa de circuito impreso entre V_{CC} y masa. El desacoplo debería realizarse muy próximo a los dispositivos que conmutan a altas velocidades o que excitan más cargas, como por ejemplo, osciladores, contadores, buffers y controladores de bus.

REVISIÓN DE LA SECCIÓN 6.10

1. Definir el término *glitch*.
2. Explicar la principal causa de los glitches en los decodificadores.
3. Definir el impulso de *validación* (strobe).

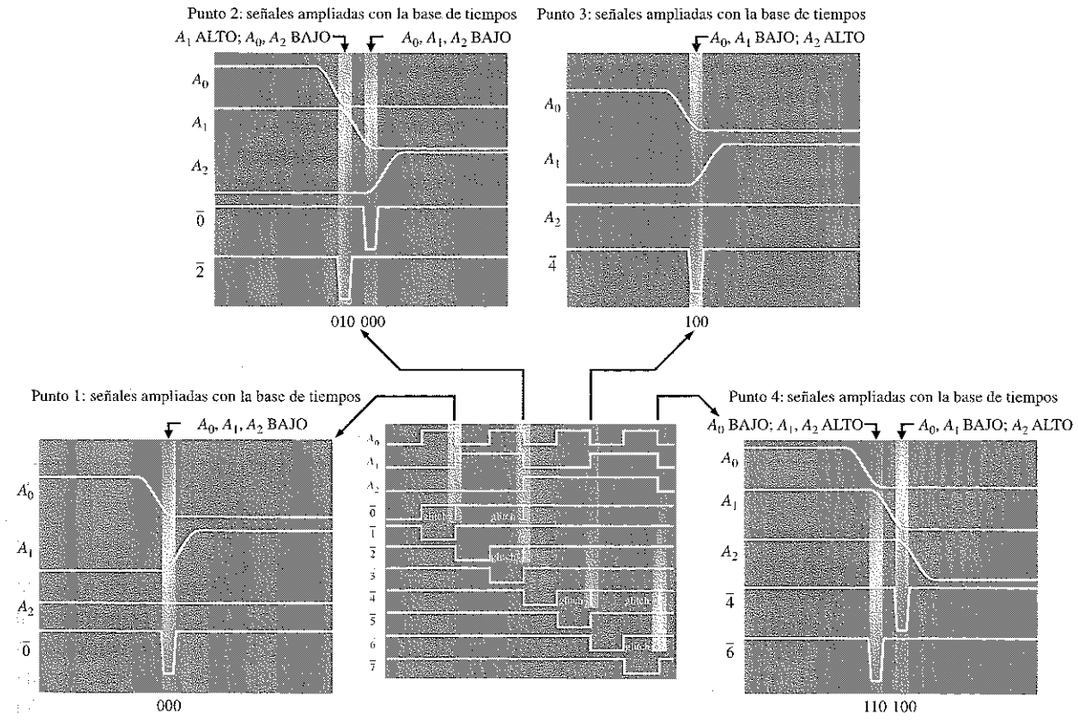


FIGURA 6.62
Señales del decodificador que muestran cómo los estados de transición de entrada producen glitches en las señales de salida.

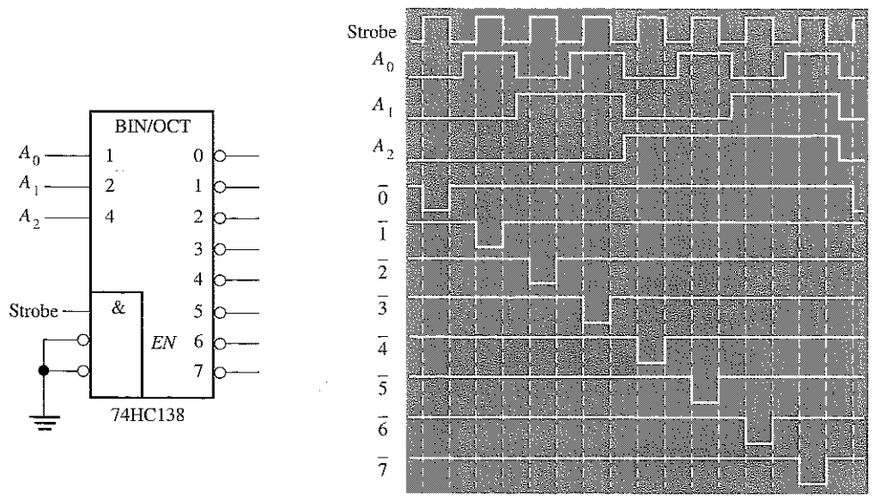


FIGURA 6.63
Aplicación de una señal de validación (strobe) para eliminar glitches en las salidas del decodificador.

6.11 ■ APLICACIÓN A LOS SISTEMAS DIGITALES

En esta aplicación, vamos a comenzar a trabajar con el sistema de control de semáforos. En esta sección se establecen los requisitos del sistema, se desarrolla un diagrama de bloques, así como un diagrama de estados para ayudar a establecer la secuencia de funcionamiento. Diseñaremos la parte del sistema que involucra lógica combinacional y se propondrán los métodos de prueba. En los capítulos 8 y 9 se tratarán los circuitos de lógica secuencial y de temporización del sistema.

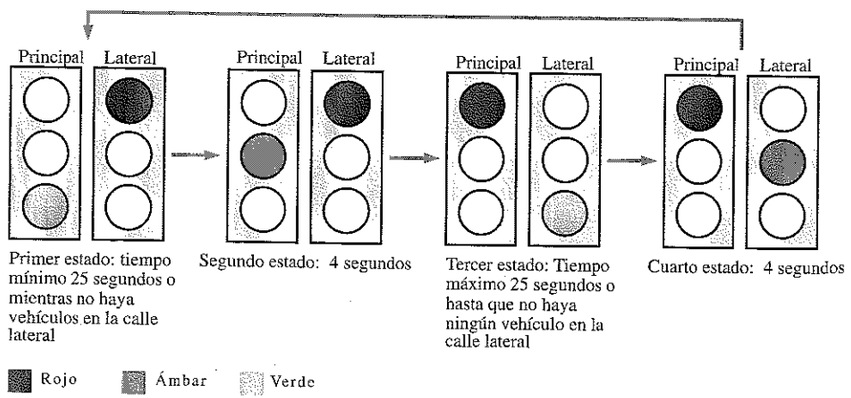
Al finalizar esta sección, el lector deberá ser capaz de:

■ Determinar las especificaciones del sistema. ■ Desarrollar un diagrama de bloques del sistema a partir de sus especificaciones básicas. ■ Explicar el propósito de un diagrama de estados. ■ Utilizar un diagrama de estados y las especificaciones del sistema para diseñar e implementar las partes lógicas del mismo. ■ Desarrollar un procedimiento de pruebas para verificar la lógica combinacional del prototipo. ■ Especificar un método sencillo para las pruebas.

Requerimientos generales del sistema

Se requiere un controlador digital para controlar un semáforo en la intersección de una calle de tráfico muy denso con una calle de tráfico moderado. La calle principal va a tener una luz verde durante un mínimo de 25 s o mientras no haya ningún vehículo en la calle perpendicular. Esta calle lateral tiene que tener la luz verde hasta que no circule ningún coche por ella, o durante un máximo de 25 s. La luz ámbar de precaución tiene que durar 4 s en los cambios de luz verde a roja en ambas calles, principal y lateral. Estos requisitos se muestran en el diagrama de la Figura 6.64.

FIGURA 6.64
Requisitos para la secuencia de luces de los semáforos.

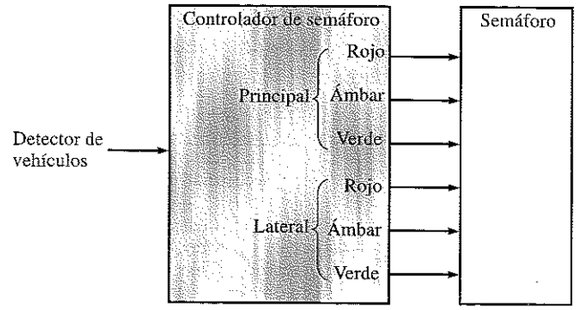


Desarrollo de un diagrama de bloques del sistema

A partir de los requisitos, se puede desarrollar un diagrama de bloques del sistema. En primer lugar, sabemos que el sistema tiene que controlar seis pares de luces diferentes. Estas son las luces roja, ámbar y verde para ambos sentidos, tanto en la calle principal

como en la lateral. También sabemos que existe una entrada externa (aparte de la alimentación) que proviene de un sensor de vehículos situado en la calle lateral. En la Figura 6.65, puede verse un diagrama de bloques mínimo que ilustra estos requisitos.

FIGURA 6.65
Diagrama de bloques mínimo del sistema.



A partir del diagrama de bloques mínimo vamos a ir entrando en los detalles. El sistema tiene cuatro estados, como se indica en la Figura 6.64, por lo que se necesita un circuito lógico para controlar la secuencia de estados (lógica secuencial). Además, se necesitan circuitos para generar los intervalos de tiempo adecuados de 25 s y 4 s, que se requieren en el sistema y para generar una señal de reloj cíclica en el sistema (circuitos de temporización). Los intervalos de tiempo (largo y corto) y el sensor de vehículos son entradas de la lógica secuencial, dado que la secuenciación de estados es una función de estas variables. Se necesitan también circuitos lógicos para determinar cuál de los cuatro estados del sistema está activo en un determinado instante de tiempo, para así generar las salidas adecuadas en las luces (decodificación de estados y lógica de salida), y para iniciar los intervalos de tiempo largo y corto. Finalmente, se necesita un circuito de interfaz para convertir los niveles lógicos de la decodificación y del circuito de salida en las tensiones y corrientes requeridas para encender cada una de las luces. La Figura 6.66 representa un diagrama de bloques más detallado que muestra estos elementos esenciales.

El diagrama de estados

Un diagrama de estados nos muestra gráficamente la secuencia de estados en un sistema y las condiciones de cada estado y de las transiciones entre cada uno de ellos. En realidad, la Figura 6.64 es, en cierta medida, un diagrama de estados, ya que muestra la secuencia de estados y las distintas condiciones.

Definición de variables. Antes de poder desarrollar un diagrama de estados, se tienen que definir las variables que determinan cómo se secuencian los estados en el sistema. Estas variables y sus símbolos son los siguientes:

- Presencia de vehículos en la calle lateral = V_S
- El temporizador de 25 segundos (largo) está *activado* = T_L
- El temporizador de 4 segundos (corto) está *activado* = T_S

El uso de variables complementadas indica la condición contraria. Por ejemplo, $\overline{V}_S$ indica que no hay ningún vehículo en la calle lateral; $\overline{T}_L$ indica que el temporizador de larga duración está desactivado, y $\overline{T}_S$ indica que el temporizador de corta duración está desactivado.

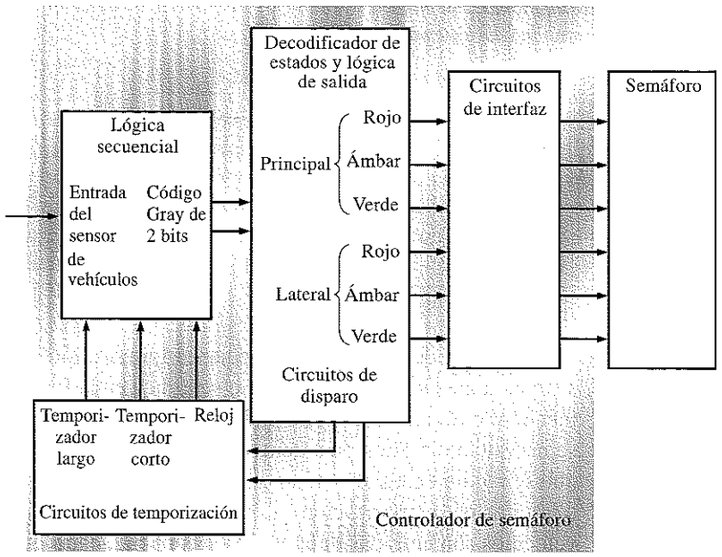


FIGURA 6.66
Diagrama de bloques del sistema en el que se indican los elementos esenciales.

Descripción del diagrama de estados. En la Figura 6.67 se muestra un diagrama de estados. Cada uno de los cuatro estados se etiqueta de acuerdo a la secuencia de 2 bits en código Gray, como se indica mediante los círculos. La flecha circular en cada estado indica que el sistema permanece en dicho estado bajo la condición definida por la variable o expresión asociada. Cada una de las flechas que van de un estado al siguiente indican un cambio de estado cuando se produce la condición definida por la variable o expresión asociada.

- ***Primer estado.*** El código Gray de este estado es 00. El semáforo de la calle principal está verde y el de la calle lateral está en rojo. El sistema permanece en este estado al menos durante 25 segundos mientras que el temporizador largo se encuentra activado o mientras no haya ningún vehículo en la calle lateral ($T_L + \overline{V}_S$). El sistema pasa al siguiente estado cuando el temporizador de 25 segundos de duración está desactivado y cuando aparece algún vehículo en la calle secundaria ($\overline{T}_L V_S$).
- ***Segundo estado.*** El código Gray de este estado es 01. El semáforo de la calle principal está en ámbar y el de la calle lateral está en rojo. El sistema permanece en este estado durante 4 segundos mientras el temporizador corto está activado (T_S) y pasa al siguiente estado cuando este temporizador se desactiva ($\overline{T}_S$).

- ***Tercer estado.*** El código Gray de este estado es 11. El semáforo de la calle principal está en rojo y el de la lateral está en verde. El sistema permanece en este estado mientras el temporizador largo está activado y hay algún vehículo en la calle lateral ($T_L V_S$). El sistema pasa al siguiente estado cuando los 25 segundos han transcurrido o cuando no hay ningún vehículo en la calle lateral, lo primero que ocurra ($\overline{T}_L + \overline{V}_S$).
- ***Cuarto estado.*** El código Gray de este estado es 10. El semáforo de la calle principal está en rojo y el de la calle lateral se encuentra en ámbar. El sistema permanece en este estado durante 4 segundos mientras el temporizador corto está activado (T_S), y vuelve al primer estado cuando este temporizador se desactiva ($\overline{T}_S$).

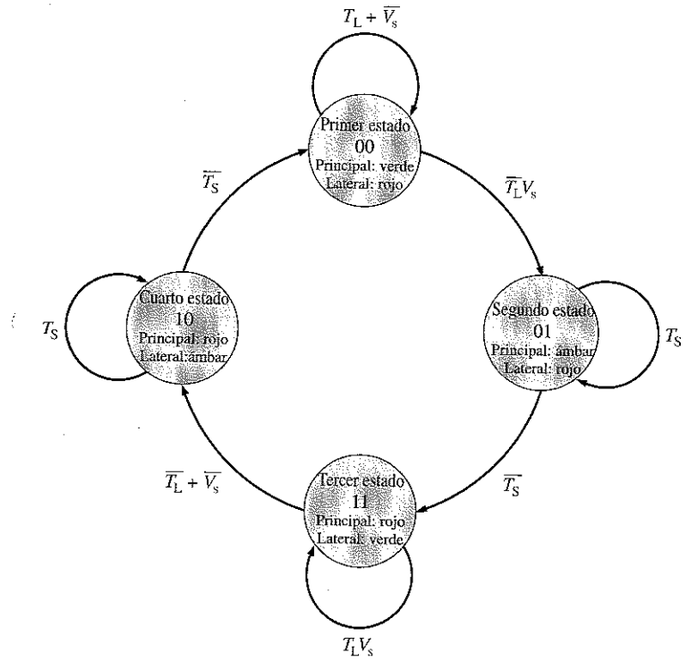


FIGURA 6.67
Diagrama de estados del sistema de control de semáforos, indicando la secuencia de código Gray.

Decodificación de estados y lógica de salida

En esta sección dedicada a las aplicaciones de sistemas nos vamos a centrar en la decodificación de estados y la lógica de salida del diagrama de bloques de la Figura 6.66. Los circuitos de temporización y de la lógica secuencial serán el tema del que tratemos en las secciones dedicadas a las aplicaciones de sistemas de los Capítulos 8 y 9.

El primer paso en el diseño de la lógica consiste en desarrollar un diagrama de bloques para la parte de la decodificación de estados y de la lógica de salida del siste-

ma. Las tres funciones que esta lógica debe realizar se definen a continuación y el diagrama resultante, con un bloque para cada una de las tres funciones, se muestra en la Figura 6.68:

- **Decodificador de estados.** Decodificar el código Gray de 2 bits de la lógica secuencial, para determinar en cuál de los cuatro estados se encuentra el sistema.
- **Lógica de salida.** Utilizar el estado decodificado para activar (mediante los circuitos de interfaz) las dos luces de los semáforos apropiadas, de las seis luces existentes.
- **Lógica del circuito de disparo.** Utilizar los estados decodificados para generar las señales que inicialicen (disparen) adecuadamente los temporizadores largo y corto.

Implementación de la lógica. El *decodificador de estados* tiene dos entradas (código Gray de 2 bits) y tiene que tener una salida para cada uno de los cuatro estados. Se necesita un decodificador de 2 líneas a 4 líneas, por lo que podríamos utilizar un 74LS139 para realizar esta tarea. El 74LS139 contiene dos decodificadores separados, pero sólo utilizaremos uno de ellos.

La *lógica de salida* tiene que tomar las cuatro salidas activas a nivel BAJO del decodificador 74LS139 y generar seis salidas que activen las luces del semáforo. En la Tabla 6.11 se muestra una tabla de verdad de la lógica del decodificador y de salida. El estado de las salidas ($\overline{SO}_1$ hasta $\overline{SO}_4$) es activo a nivel BAJO (0) y se requiere que las salidas para la activación de las luces, a partir de la lógica de salida, sean activas a nivel BAJO (0).

La *lógica del circuito de disparo* produce dos salidas. La salida para el circuito de temporización de 25 s (temporizador largo) da lugar a una transición de nivel BAJO a nivel ALTO, cuando el sistema pasa a los estados primero o tercero. La salida para disparar el circuito temporizador de 4 s (temporizador corto) da lugar a una transición de nivel BAJO a nivel ALTO cuando el sistema pasa a los estados segundo o cuarto. Estas observaciones se pueden comprobar examinando los requisitos del sistema y el diagrama de estados de la Figura 6.67.

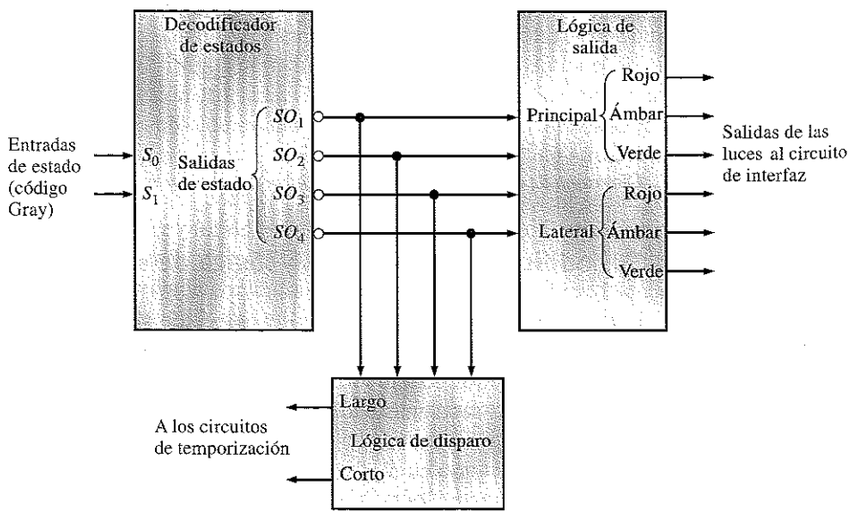


FIGURA 6.68
Diagrama de bloques para la decodificación de estados y la lógica de salida.

TABLA 6.11
Tabla de verdad de la lógica de decodificación y de salida.

Entradas de estado		Salidas de estado				Salidas de las luces						Salidas del circuito de disparo	
S_1	S_0	SO_1	SO_2	SO_3	SO_4	$\overline{PR}$	$\overline{PA}$	$\overline{PV}$	$\overline{LR}$	$\overline{LA}$	$\overline{LV}$	*Largo*	*Corto*
0	0	0	1	1	1	1	1	0	0	1	1	1	0
0	1	1	0	1	1	1	0	1	0	1	1	0	1
1	1	1	1	0	1	0	1	1	1	1	0	1	0
1	0	1	1	1	0	0	1	1	1	0	1	0	1

Las salidas de estados son activas a nivel BAJO y las salidas de las luces son activas a nivel ALTO. *PR:* principal rojo; *LR:* lateral rojo; etc.

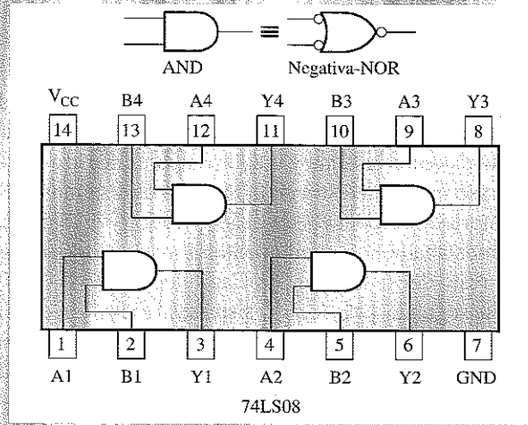
■ PRÁCTICAS DE SISTEMAS

La primera práctica de sistemas implica el diseño del decodificador de estados y de la lógica de salida del sistema. Conviene esforzarse en diseñar el diagrama lógico requerido en esta parte del sistema y luego comparar nuestros resultados con la parte del circuito impreso que se muestra en la Práctica de sistemas 2. Sólo se muestra la parte del circuito impreso que contiene el decodificador y la lógica de salida. Téngase en cuenta que iremos añadiendo componentes a nuestro diagrama lógico a medida que estudiemos el resto del sistema en los Capítulos 8 y 9.

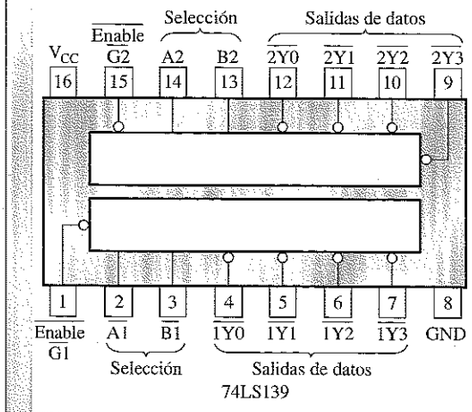
En la segunda práctica de sistemas, todas las conexiones a V_{CC} y a masa se encuentran en la parte posterior del circuito impreso. Las interconexiones de la parte posterior del circuito se realizan a través de los contactos de la cara de componentes.

■ PRÁCTICAS DE SISTEMAS 1: Diseño

- ***Actividad 1***. A partir de la información de la Tabla 6-11 y del diagrama de pines extraído de la hoja de características de un 74LS08, que se muestra más abajo, diseñar la lógica de salida necesaria para generar las seis salidas de las luces al circuito de interfaz. Dibujar el diagrama lógico. *Consejo: una puerta AND puede ser considerada como una puerta negativa-NOR en la que un nivel BAJO en cualquiera de sus entradas o en ambas produce una salida a nivel BAJO, como se muestra más abajo.*
- ***Actividad 2***. Utilizando el diagrama de estados y el 74LS08, diseñar la lógica del circuito de disparo para cumplir los requisitos anteriormente establecidos. Añadir esto al diagrama lógico de la actividad 1.



- ***Actividad 3***. Combinar los circuitos de las actividades 1 y 2 con el decodificador 74LS139, cuya tabla de verdad y diagrama de pines, obtenidos de su hoja de características, se muestran más abajo, para crear un diagrama lógico completo del decodificador de estados y de la lógica de salida del sistema. Asegúrese de que el diagrama incluye la numeración de los pines.



Entradas			Salidas			
Habilitación	Selección					
$\overline{G}$	B	A	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

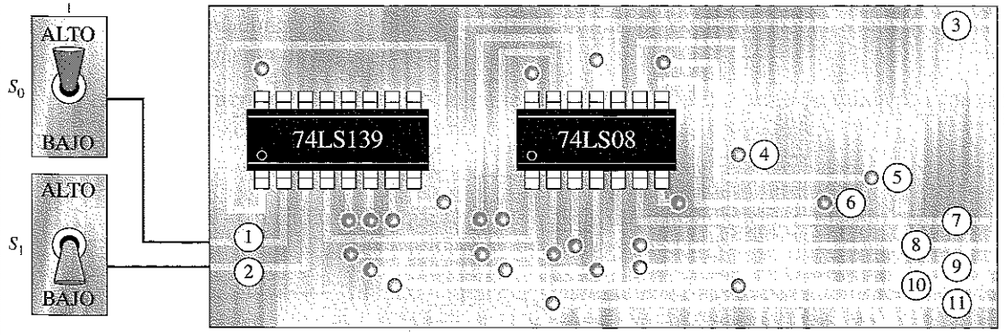
NOTA: H = Nivel alto; L = Nivel bajo; X = Indiferente

■ PRÁCTICAS DE SISTEMAS 2: Verificación y pruebas

- ***Actividad 1***. Comparar el diagrama lógico que ha desarrollado en el Práctica de sistemas 1 con la parte mostrada de la tarjeta de circuito impreso, y verificar que concuerdan.
- ***Actividad 2***. Identificar la función de cada uno de los puntos de la tarjeta de circuito impreso marcados con un círculo y numerados.
- ***Actividad 3***. Utilizando los interruptores conectados tal y como se muestra para simular las entradas de código Gray de la parte secuencial del sistema, desarrollar un procedimiento de pruebas para esta parte del sistema.
- ***Actividad 4***. Cuando los interruptores se sitúan en cada una de las posiciones mostradas en la tabla, determinar a partir de las indicaciones de los puntos de prueba representados por los círculos numerados si el circuito está funcionando adecuadamente en cada caso y, de no ser así, cuál es la causa más probable de error. Analizar cada caso por separado. Cada uno de los casos es independiente de los restantes indicados en la tabla.

La lógica de la tarjeta de circuito impreso para el decodificador de estados y la lógica de salida se encuentra en el archivo SA06.EWB del CD-ROM EWB. Utilícese el EWB para simular una prueba de este circuito.

S_1	S_0	Caso	③	④	⑤	⑥	⑦	⑧	⑨	⑩	⑪
ALTO BAJO	ALTO BAJO	1	BAJO(L)	ALTO(H)	ALTO(H)	BAJO(L)	ALTO(H)	BAJO(L)	ALTO(H)	ALTO(H)	ALTO(H)
		2	BAJO(L)	ALTO(H)	ALTO(H)	BAJO(L)	ALTO(H)	ALTO(H)	ALTO(H)	ALTO(H)	ALTO(H)
ALTO BAJO	ALTO BAJO	3	ALTO(H)	ALTO(H)	BAJO(L)	ALTO(H)	ALTO(H)	BAJO(L)	ALTO(H)	ALTO(H)	BAJO(L)
		4	ALTO(H)	ALTO(H)	ALTO(H)	ALTO(H)	ALTO(H)	BAJO(L)	ALTO(H)	ALTO(H)	BAJO(L)
ALTO BAJO	ALTO BAJO	5	ALTO(H)	BAJO(L)	ALTO(H)	BAJO(L)	BAJO(L)	ALTO(H)	ALTO(H)	BAJO(L)	ALTO(H)
		6	ALTO(H)	ALTO(H)	ALTO(H)	ALTO(H)	ALTO(H)	ALTO(H)	ALTO(H)	ALTO(H)	ALTO(H)
ALTO BAJO	ALTO BAJO	7	ALTO(H)	ALTO(H)	BAJO(L)	ALTO(H)	BAJO(L)	ALTO(H)	BAJO(L)	ALTO(H)	ALTO(H)
		8	ALTO(H)	ALTO(H)	BAJO(L)	ALTO(H)	ALTO(H)	ALTO(H)	BAJO(L)	ALTO(H)	ALTO(H)



REVISIÓN DE LA SECCIÓN 6.11

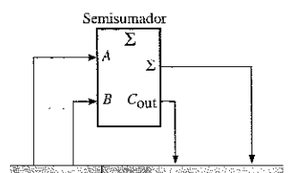
1. Enumerar la secuencia de estados en las entradas del decodificador de estados.
2. Explicar cómo las puertas AND empleadas para la lógica de salida y para la lógica del circuito de disparo, en las prácticas de sistemas, actúan como puertas negativa-NOR.
3. ¿Cuándo ocurren las transiciones de salida del circuito de disparo para el temporizador largo? ¿Y para el temporizador corto?

En este punto puede estudiarse el Capítulo 15, "Tecnología de circuitos integrados", parcial o totalmente. El examen de este capítulo no es un prerrequisito para abordar cualquier otro tema tratado en este texto, y su omisión total o parcial no afectará al resto de los temas.

RESUMEN

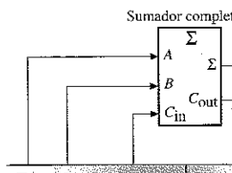
- El funcionamiento del semisumador y del sumador completo se muestran en la Figura 6.69.
- En la Figura 6.70 se muestran los símbolos lógicos y la numeración de pines de los circuitos integrados empleados en este capítulo. La designación de pines puede variar dependiendo del fabricante.

Semisumador



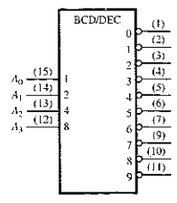
Entradas A	Entradas B	Salida de acarreo Cout	Suma Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Sumador completo

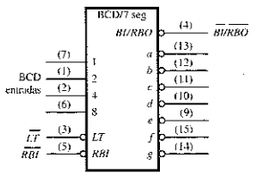


Entradas A	Entradas B	Entrada de acarreo Cin	Salida de acarreo Cout	Suma Σ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

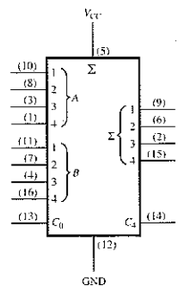
FIGURA 6.69



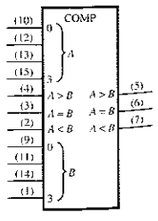
Decodificador BCD-decimal 74HC42



Controlador/decodificador BCD-7 segmentos 74LS47/74LS48

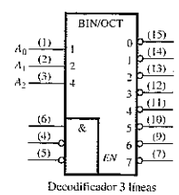


Sumador de bits 74LS583A

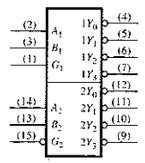


Comparador de magnitud de 4 bits 74HC85

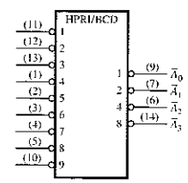
FIGURA 6.70



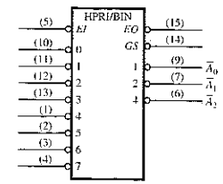
Decodificador 3 líneas a 8 líneas 74LS138



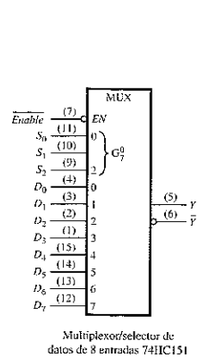
Decodificador 2 líneas a 4-líneas 74LS139



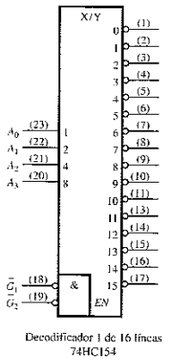
Codificador con prioridad decimal-BCD 74HC147



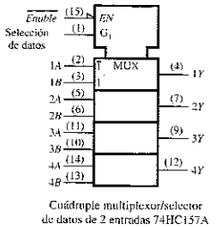
Codificador octa-binario 74F148



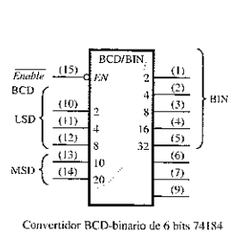
Multiplexor/selector de datos de 8 entradas 74HC151



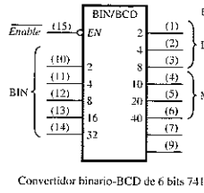
Decodificador 1 de 16 líneas 74HC154



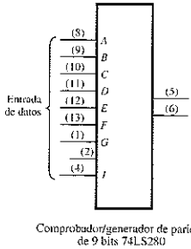
Cuádruple multiplexor/selector de datos de 2 entradas 74HC157A



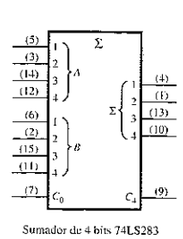
Convertidor BCD-binario de 6 bits 74184



Convertidor binario-BCD de 6 bits 74185



Comprobador/generador de paridad de 9 bits 74LS280



Sumador de 4 bits 74LS283

FIGURA 6.70 *(Continuación)*

PALABRAS CLAVE

Estos términos también se encuentran en el glosario del final del libro.

Acarreo anticipado Método de suma binaria en que los acarreos de las etapas sumadoras precedentes se anticipan, eliminando de esta manera los retardos en la propagación de los acarreos.

Acarreo serie Método de suma binaria en el que el acarreo de salida de cada sumador se convierte en el acarreo de entrada del sumador siguiente.

Aceptor Dispositivo receptor en un bus.

Bit de paridad Bit que se añade a cada grupo de bits de información para hacer que el número de 1s sea par o impar en dicho grupo de bits.

Codificador Circuito digital que convierte la información a un formato codificado.

Codificador con prioridad Codificador en el que sólo se codifica el dígito de entrada de valor más alto, ignorándose cualquier otra entrada activa.

Conexión en cascada Conectar la salida de un dispositivo a la entrada de un dispositivo similar, permitiendo a uno de los dispositivos excitar a otro, para aumentar la capacidad de operación.

Convertidor de código Dispositivo digital que convierte información codificada de una manera en otra codificación distinta.

Decodificador Circuito digital que convierte la información codificada en un formato más familiar o no codificado.

Demultiplexor (DEMUX) Circuito que conmuta los datos digitales desde una línea de entrada a varias líneas de salida.

Generación de acarreo El proceso mediante el que se produce un acarreo de salida en un sumador completo cuando las dos entradas son 1.

Glitch Pico de tensión o de corriente de corta duración, no deseado y generalmente producido de forma no intencionada.

Multiplexor (MUX) Circuito que conmuta los datos digitales de distintas líneas de entrada a una única línea de salida.

Nibble Un grupo de cuatro bits.

Notación de dependencia Sistema de notación para los símbolos lógicos que especifica las relaciones de entrada y salida, definiendo una determinada función de forma completa; parte integral del estándar ANSI/IEEE 91-1984.

Paridad impar Condición de poseer un número impar de unos en cada grupo de bits.

Paridad par Condición por la que se tiene un número par de unos en cada grupo de bits.

Propagación de acarreo Proceso mediante el que se propaga el acarreo de entrada de un sumador completo hasta que se convierte en acarreo de salida, cuando uno o los dos bits de entrada son 1 y el acarreo de entrada es 1.

Resistencia de pull-up Resistencia que se utiliza para mantener un determinado punto de un circuito a nivel ALTO cuando se encuentra en estado inactivo.

Selector de datos Circuito que selecciona los datos de varias entradas a un tiempo según una secuencia y los coloca en la salida; también se denomina multiplexor.

Semisumador Circuito digital que suma dos bits y produce una suma y un acarreo de salida. No puede manejar acarreos de entrada.

Sumador completo Circuito digital que suma dos bits y un acarreo de entrada para producir una suma y un acarreo de salida.

Supresión de cero Proceso de eliminar los ceros anteriores o posteriores en una presentación en display digital.

Validación Proceso por el que se utiliza un pulso para muestrear la aparición de un suceso, en un determinado instante de tiempo relativo a dicho suceso.

■ AUTOTEST

1. Un semisumador se caracteriza por tener:
   (a) dos entradas y dos salidas
   (b) tres entradas y dos salidas
   (c) dos entradas y tres salidas
   (d) dos entradas y una salida
2. Un sumador completo se caracteriza por tener:
   (a) dos entradas y dos salidas
   (b) tres entradas y dos salidas
   (c) dos entradas y tres salidas
   (d) dos entradas y una salida
3. Las entradas de un sumador completo son $A = 1$, $B = 1$ y $C_{in} = 0$. Sus salidas son:
   (a) $\Sigma = 1$, $C_{out} = 1$
   (b) $\Sigma = 1$, $C_{out} = 0$
   (c) $\Sigma = 0$, $C_{out} = 1$
   (d) $\Sigma = 0$, $C_{out} = 0$
4. Un sumador en paralelo de 4 bits puede sumar:
   (a) dos números binarios de 4 bits
   (b) dos números binarios de 2 bits
   (c) cuatro bits a la vez
   (d) una secuencia de cuatro bits
5. El 74LS83A es un ejemplo de sumador en paralelo de 4 bits. Para ampliar este dispositivo a un sumador de 8 bits, hay que:
   (a) utilizar cuatro sumadores no interconectados
   (b) utilizar dos sumadores y conectar las salidas de la suma de uno de ellos a las entradas de datos del otro
   (c) utilizar ocho sumadores no interconectados
   (d) utilizar dos sumadores con la salida de acarreo de uno de ellos conectada a la entrada de acarreo del otro
6. Si un comparador de magnitud 74HC85 tiene $A = 1011$ y $B = 1001$ en sus entradas, las salidas son:
   (a) $A > B = 0, A < B = 1, A = B = 0$
   (b) $A > B = 1, A < B = 0, A = B = 0$
   (c) $A > B = 1, A < B = 1, A = B = 0$
   (d) $A > B = 0, A < B = 0, A = B = 1$
7. Si un decodificador de 4 líneas a 16 líneas con salidas activas a nivel bajo muestra un nivel bajo en la salida decimal 12, ¿cuáles son sus entradas?
   (a) $A_3A_2A_1A_0 = 1010$ **(b)** $A_3A_2A_1A_0 = 1110$
   (c) $A_3A_2A_1A_0 = 1100$ **(d)** $A_3A_2A_1A_0 = 0100$
8. Un decodificador BCD a 7 segmentos tiene 0100 en sus entradas. Las salidas activas serán:
   (a) *a, c, f, g* **(b)** *b, c, f, g*
   (c) *b, c, e, f* **(d)** *b, d, e, g*
9. Si un codificador de prioridad octal-binario tiene sus entradas 0, 2, 5 y 6 en un nivel activo, la salida binaria activa a nivel alto será:
   (a) 110 **(b)** 010 **(c)** 101 **(d)** 000
10. Por regla general, un multiplexor tiene:
    (a) una entrada de datos, varias salidas de datos y entradas de selección
    (b) una entrada de datos, una salida de datos y una entrada de selección
    (c) varias entradas de datos, varias salidas de datos y entradas de selección
    (d) varias entradas de datos, una salida de datos y entradas de selección

11. Los selectores de datos son, básicamente, lo mismo que los:

(a) decodificadores
(b) demultiplexores
(c) multiplexores
(d) codificadores

12. ¿Cuáles de los siguientes códigos poseen paridad par?

(a) 10011000 **(b)** 01111000
(c) 11111111 **(d)** 11010101
(e) todos **(f)** respuestas (b) y (c)

■ PROBLEMAS

SECCIÓN 6.1 Sumadores básicos

1. Para el sumador completo de la Figura 6.4, determinar el estado lógico (1 o 0) a la salida de cada puerta para las siguientes entradas:

(a) $A = 1, B = 1, C_{in} = 1$ **(b)** $A = 0, B = 1, C_{in} = 1$ **(c)** $A = 0, B = 1, C_{in} = 0$

2. ¿Cuáles serían las entradas que producirían en un sumador completo las siguientes salidas?

(a) $\Sigma = 0, C_{out} = 0$ **(b)** $\Sigma = 1, C_{out} = 0$
(c) $\Sigma = 1, C_{out} = 1$ **(d)** $\Sigma = 0, C_{out} = 1$

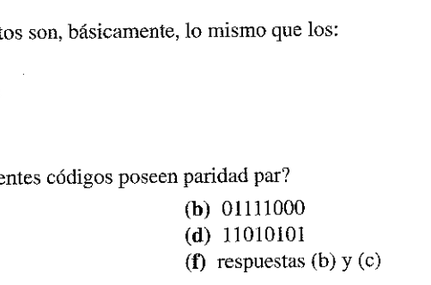
3. Determinar las salidas de un sumador completo para cada una de las siguientes entradas:

(a) $A = 1, B = 0, C_{in} = 0$ **(b)** $A = 0, B = 0, C_{in} = 1$
(c) $A = 0, B = 1, C_{in} = 1$ **(d)** $A = 1, B = 1, C_{in} = 1$

SECCIÓN 6.2 Sumadores binarios en paralelo

4. Para el sumador en paralelo de la Figura 6.71, determinar la suma completa mediante el análisis del funcionamiento lógico del circuito. Comprobar el resultado sumando manualmente los dos números de entrada.

FIGURA 6.71



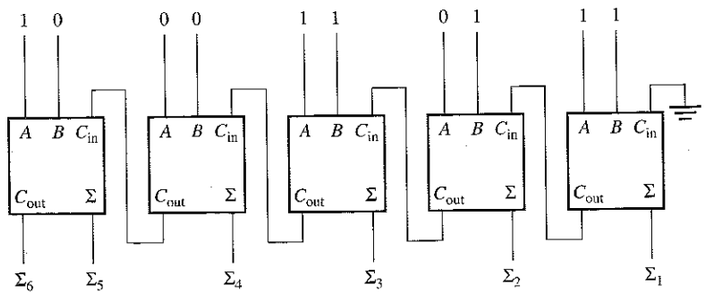
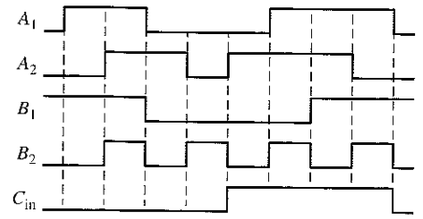


FIGURA 6.72

FIGURA 6.73



5. Repetir el problema 4 para el circuito y las condiciones de entrada de la Figura 6.72.

6. Las formas de onda de entrada de la Figura 6.73 se aplican a un sumador de 2 bits. Determinar, mediante un cronograma, las señales correspondientes a la suma y a la salida de acarreo, en función de las entradas.

7. Las siguientes secuencias de bits (el bit de la derecha es el primero) se aplican a las entradas de un sumador 74LS83A. Determinar la secuencia de bits resultante en cada salida.

A_1	1001
A_2	1110
A_3	0000
A_4	1011
B_1	1111
B_2	1100
B_3	1010
B_4	0010

8. En las pruebas de un sumador completo de 4 bits 74LS83A, se observan los siguientes niveles de tensión en sus pines: 1-BAJO, 2-ALTO, 3-ALTO, 4-ALTO, 6-ALTO, 7-ALTO, 8-BAJO, 9-BAJO, 10-BAJO, 11-BAJO, 13-BAJO, 14-ALTO, 15-BAJO y 16-ALTO. Determinar si el circuito integrado funciona correctamente.

SECCIÓN 6.3 Comparadores

9. Se aplican las formas de onda mostradas en la Figura 6.74 a las entradas del comparador. Determinar la señal de salida ($A = B$).

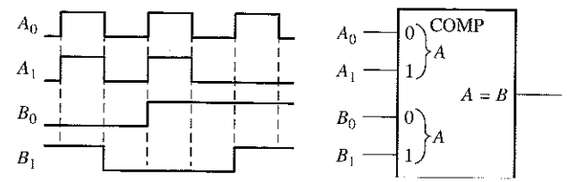


FIGURA 6.74

10. Para el comparador de 4 bits de la Figura 6.75, dibujar cada forma de onda de salida para las entradas que se muestran. Las salidas son activas a nivel ALTO.

11. Para los siguientes grupos de números binarios, determinar los estados de salida para el comparador de la Figura 6.18.

(a) $A_3A_2A_1A_0 = 1100$
$B_3B_2B_1B_0 = 1001$

(b) $A_3A_2A_1A_0 = 1000$
$B_3B_2B_1B_0 = 1011$

(c) $A_3A_2A_1A_0 = 0100$
$B_3B_2B_1B_0 = 0100$

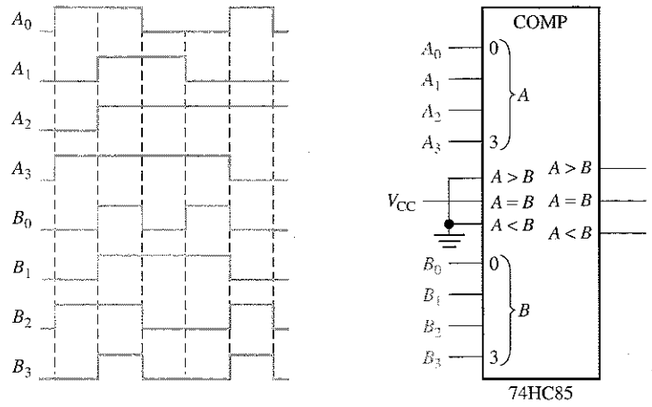


FIGURA 6.75

SECCIÓN 6.4 Decodificadores

12. Cuando en la salida de cada puerta de decodificación de la Figura 6.76 hay un nivel ALTO, ¿cuál es el código binario que aparece en sus entradas? El bit más significativo (MSB) es A_3.

13. ¿Cuál es la lógica de decodificación para cada uno de los siguientes códigos, si se requiere una salida activa a nivel ALTO (1)?

(a) 1101 **(b)** 1000 **(c)** 11011 **(d)** 11100
(e) 101010 **(f)** 111110 **(g)** 000101 **(h)** 1110110

14. Resolver el problema 13, suponiendo que se requiere una salida activa a nivel BAJO (0).

15. Se desea detectar únicamente la presencia de los códigos 1010, 1100, 0001 y 1011. Para indicar la presencia de dichos códigos se requiere una salida activa a nivel ALTO. Desarrollar la lógica de decodificación mínima necesaria que tenga una única salida que indique cuándo cualquiera de estos códigos se encuentra en las entradas. Para cualquier otro código, la salida ha de ser un nivel BAJO.

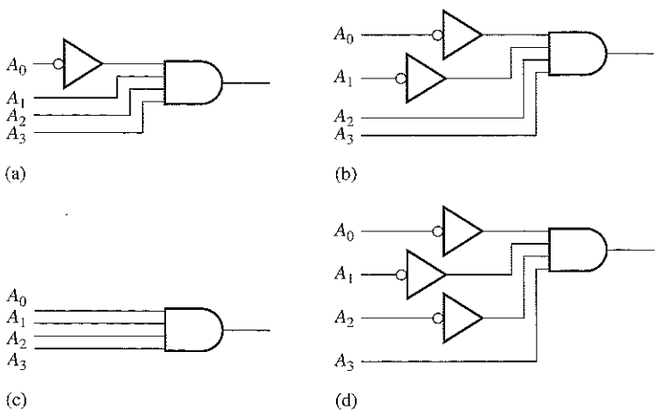


FIGURA 6.76

16. Si se aplican las formas de onda de entrada a la lógica de decodificación de la Figura 6.77, dibujar las formas de onda de salida en función de dichas entradas.

17. Se aplican secuencialmente números BCD al decodificador BCD-decimal de la Figura 6.78. Dibujar un diagrama de tiempos que muestre cada salida en relación con el resto de las señales de salida y con las de entrada.

18. Un decodificador/excitador de 7-segmentos controla el display de la Figura 6.79. Si se aplican las formas de onda de entrada que se muestran, determinar la secuencia de dígitos que aparece en el display.

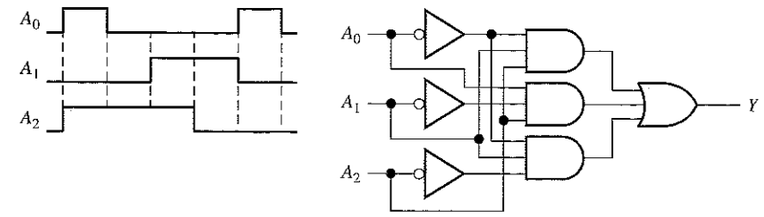


FIGURA 6.77

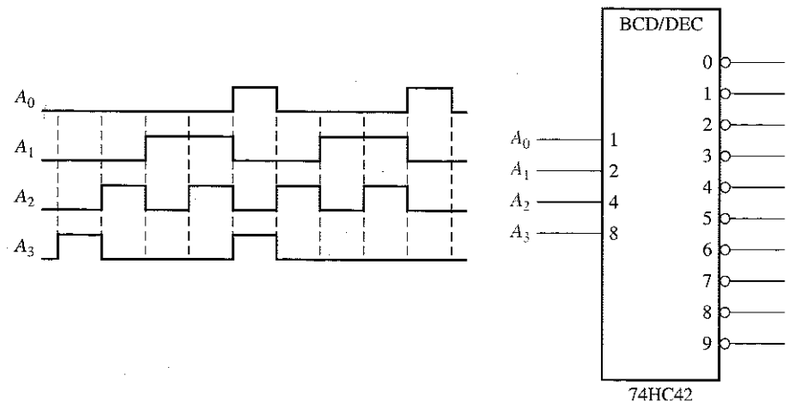


FIGURA 6.78

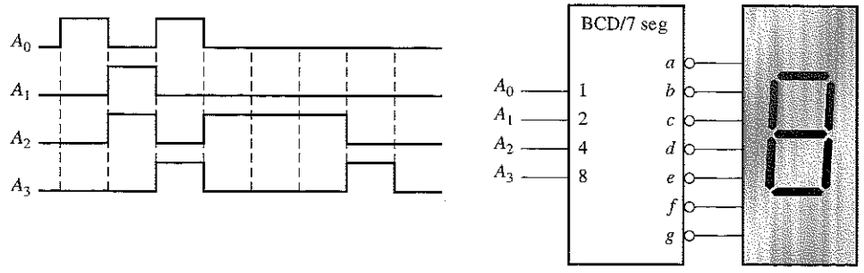


FIGURA 6.79

SECCIÓN 6.5 Codificadores

19. Suponer que el codificador lógico decimal-BCD de la Figura 6.34 tiene las entradas 3 y 9 a nivel ALTO. ¿Cuál es el código de salida? ¿Es éste un código BCD (8421) válido?

20. Un decodificador 74HC147 tiene niveles BAJOS de tensión en sus pines 2, 5 y 12. ¿Qué código BCD aparece en las salidas si todas las demás entradas están a nivel ALTO?

SECCIÓN 6.6 Convertidores de código

21. Convertir primero a BCD los siguientes números decimales y luego, aplicar los dígitos BCD al convertidor BCD-binario de la Figura 6.39(a) e indicar las salidas binarias resultantes.

(a) 2 **(b)** 8 **(c)** 13 **(d)** 26 **(e)** 33

22. Explicar la lógica requerida para convertir a código Gray un número binario de 10 bits, y utilizar esta lógica para convertir los siguientes números binarios:

(a) 1010101010 **(b)** 1111100000
(c) 0000001110 **(d)** 1111111111

23. Explicar la lógica requerida para convertir a binario un código Gray de 10 bits y utilizar esta lógica para convertir a binario los siguientes códigos Gray:

(a) 1010000000 **(b)** 0011001100
(c) 1111000111 **(d)** 0000000001

SECCIÓN 6.7 Multiplexores (selectores de datos)

24. En el demultiplexor de la Figura 6.80, determinar la salida para los siguientes estados de entrada:

$D_0 = 0, D_1 = 1, D_2 = 1, D_3 = 0, S_0 = 1, S_1 = 0$

25. Si las entradas de selección de datos del multiplexor de la Figura 6.80 se secuencian tal y como se muestra en las formas de onda de la Figura 6.81, determinar la forma de onda de salida para los datos de entrada del problema 24.

26. Las formas de onda mostradas en la Figura 6.82 se aplican a las entradas de un multiplexor de ocho entradas 74HC151. Dibujar la señal de salida *Y*.

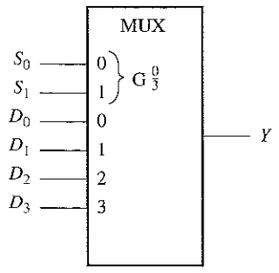


FIGURA 6.80

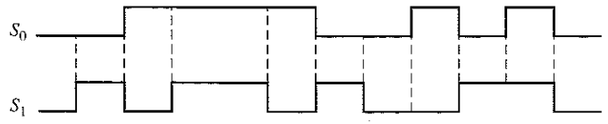


FIGURA 6.81

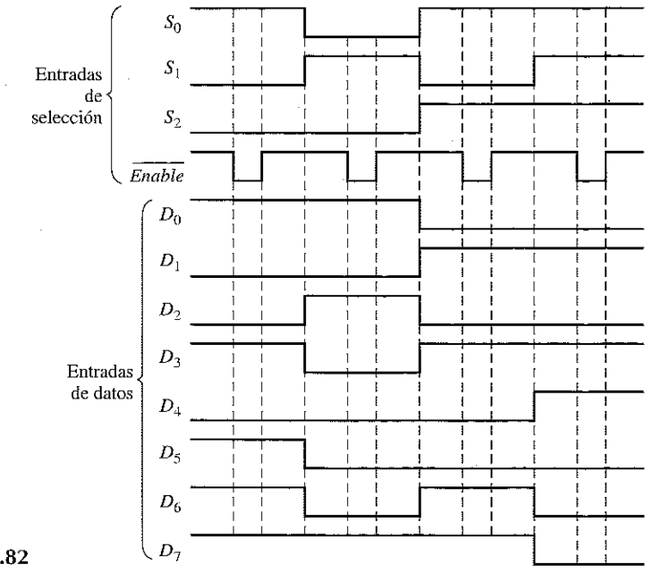


FIGURA 6.82

SECCIÓN 6.8 Demultiplexores

27. Desarrollar el diagrama de tiempos completo (entradas y salidas) de un 74HC154 utilizado en una aplicación de demultiplexación en el que las entradas son las siguientes: las entradas de selección de datos toman, de forma repetitiva y secuencialmente, los valores generados por un contador binario que comienza en 0000, y la entrada de datos es una cadena de datos serie, en BCD, que representan al número decimal 2468. El dígito menos significativo (8) es el primero de la secuencia, con el bit menos significativo en primer lugar, y deberá aparecer en los cuatro primeros bits de la salida.

SECCIÓN 6.9 Generadores / Comprobadores de paridad

28. Se aplican las formas de onda de la Figura 6.83 al circuito de paridad de 4 bits. Determinar las señales de salida en función de las entradas. ¿Durante cuántos periodos de bit ocurre la paridad par y cómo se indica? El diagrama de tiempos incluye ocho periodos de bit.

29. Determinar las salidas $\sum$ Impar y $\sum$ Par de un generador/comprobador de paridad 74LS280 de 9 bits, para las entradas de la Figura 6.84. Utilizar la tabla de verdad de la Figura 6.58.

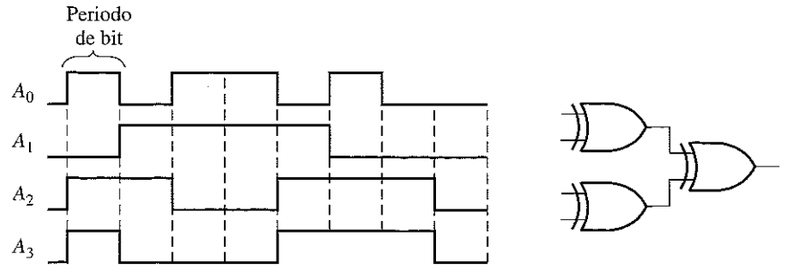


FIGURA 6.83

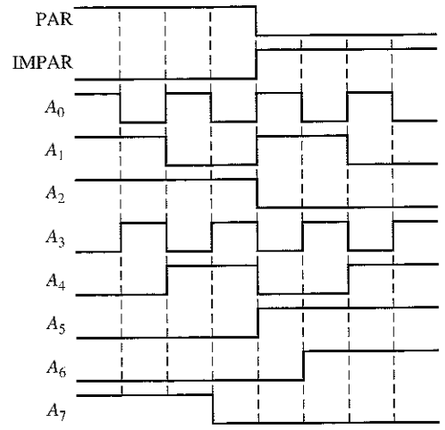


FIGURA 6.84

SECCIÓN 6.10 Localización de averías

30. El sumador completo de la Figura 6.85 se prueba bajo todas las condiciones de entrada posibles, con las señales de entrada indicadas. A partir de la observación de las señales Σ y C_{out}, ¿funciona correctamente? Si la respuesta es no, ¿cuál es la causa más probable de fallo?

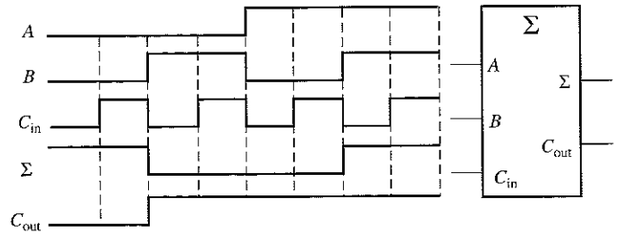


FIGURA 6.85

31. Enumerar los posibles fallos de cada codificador/display de la Figura 6.86.

32. Desarrollar un procedimiento de pruebas sistemático para verificar el funcionamiento completo del codificador de teclado de la Figura 6.38.

33. Hay que probar el convertidor BCD-binario formado por cuatro sumadores que se muestra en la Figura 6.87. El sistema requiere la aplicación secuencial de números BCD, comenzando por 0_{10}, para comprobar que la salida binaria es la correcta. ¿Qué síntoma o síntomas aparecerían en las salidas binarias si ocurrieran cada uno de los siguientes fallos? ¿Cuál es el número BCD para el que se detecta por *primera vez* cada error?

(a) La entrada A_1 está en circuito abierto (sumador superior).
(b) C_{out} está en circuito abierto (sumador superior).
(c) La salida Σ_4 está cortocircuitada a masa (sumador superior).
(d) La salida 32 está cortocircuitada a masa (sumador inferior).

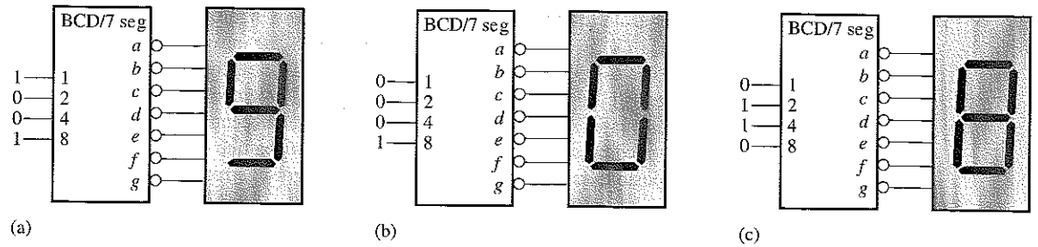


FIGURA 6.86

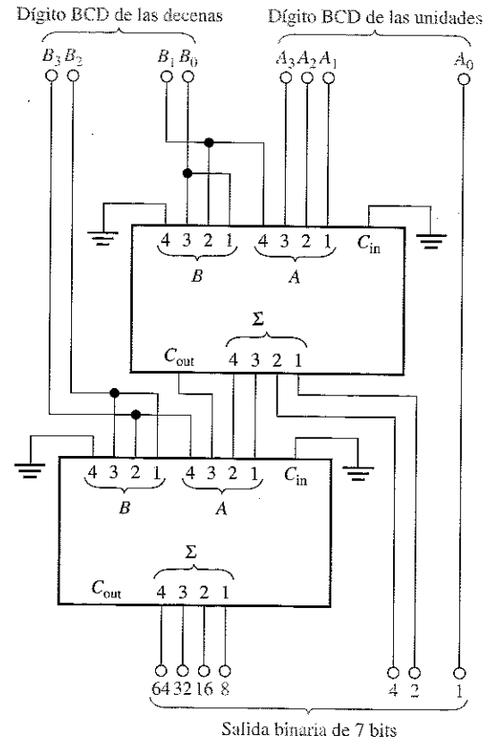


FIGURA 6.87

34. En el display con multiplexación de la Figura 6.51, determinar la causa (o las causas) más probable para cada uno de los siguientes síntomas:

(a) El display del dígito *B* (el más significativo) no se enciende.
(b) Ninguno de los displays de 7 segmentos se enciende.
(c) El segmento *f* de ambos displays aparece encendido siempre.
(d) Hay un parpadeo visible en los displays.

35. Desarrollar un procedimiento sistemático para probar exhaustivamente el CI selector de datos 74HC151.

36. Durante las pruebas del sistema de transmisión de datos de la Figura 6.59, se aplica un código a las entradas D_0 a D_6 que contiene un número impar de 1s. Se introduce deliberadamente un único bit erróneo en la línea de transmisión serie entre el multiplexor y el demultiplexor, pero el sistema no detecta el error (salida de error = 0). Tras algún tiempo de investigación, se verifican las entradas con el comprobador de paridad par y se encuentra que en D_0 a D_6 hay un número par de 1s, como se esperaba, y también se comprueba que el bit de paridad D_7 es 1. ¿Cuáles son las posibles razones de que el sistema no indique el error?

37. Describir de forma general cómo probaríamos el sistema de transmisión de datos de la Figura 6.59 y especificar un método de introducción de errores de paridad.

SECCIÓN 6.11 Aplicación a los sistemas digitales

38. El bloque de la lógica de salida se implementa en el banco de trabajo mediante un 74LS08 con puertas AND operando como puertas negativa-NOR. Utilizar un 74LS00 (puertas NAND cuádruples) y cualquier otro dispositivo que sea necesario para generar salidas activas a nivel ALTO.

39. Implementar la lógica de salida si se necesitan salidas activas a nivel BAJO.

Problemas especiales de diseño

40. Modificar el diseño del sistema de multiplexación del display de 7 segmentos de la Figura 6.52 para permitir visualizar dos dígitos adicionales.

41. Utilizando la Tabla 6.2, escribir las expresiones de suma de productos para Σ y C_{out} de un sumador completo. Utilizar un mapa de Karnaugh para minimizar las expresiones y luego implementarlas empleando inversores y lógica AND-OR. Indicar cómo se puede reemplazar la lógica AND-OR con selectores de datos 74HC151.

42. Implementar la función lógica especificada en la Tabla 6.12 utilizando un selector de datos 74HC151.

TABLA 6.12

Entradas				Salida
A_3	A_2	A_1	A_0	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

43. Utilizando dos de los módulos sumadores de 6 posiciones de la Figura 6.14, diseñar un sistema de votación de 12 posiciones.

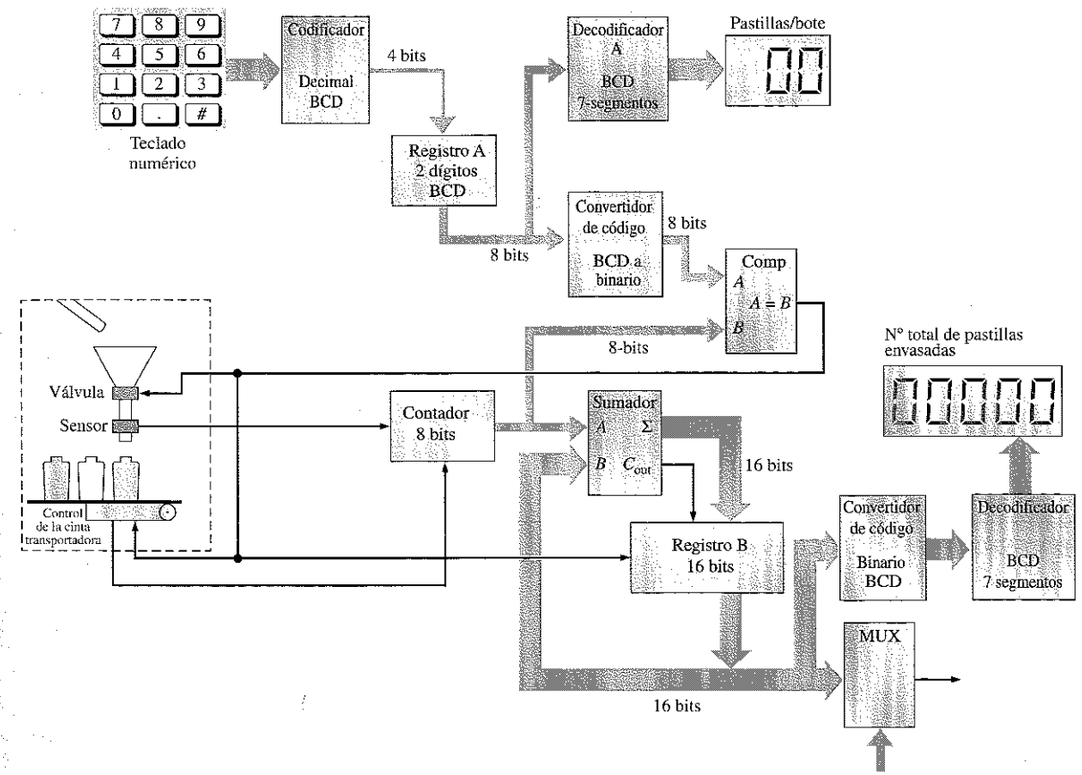


FIGURA 6.88

44. El bloque sumador del sistema de control y recuento de pastillas de la Figura 6.88 realiza la suma del número binario de 8 bits del contador y del número binario de 16 bits del registro B. El resultado de la suma se almacena en el registro B. Utilizar circuitos 74LS283 para implementar esta función y dibujar un diagrama lógico completo que incluya la numeración de los pines. Revisar el funcionamiento del sistema en los Capítulos 1 y 2.

45. El bloque comparador del sistema de control y recuento de pastillas de la Figura 6.88 se implementó en el Capítulo 3 utilizando puertas lógicas discretas. Ahora estamos preparados para utilizar dispositivos MSI con el fin de simplificar el diseño. El comparador compara el número binario de 8 bits (en realidad sólo se requieren 7 bits) del convertidor BCD-binario con el número binario de 8 bits del contador. Utilizar circuitos 74HC85 para implementar esta función y dibujar un diagrama lógico completo que incluya la numeración de pines.

46. Se utilizan dos decodificadores BCD-7 segmentos en el sistema de control y recuento de la Figura 6.88. Uno de ellos se requiere para controlar el display de dos dígitos *pastillas/bote*, y el otro para controlar el display de 5 dígitos *número total de pastillas envasadas*. Utilizar circuitos 74LS47 para implementar cada decodificador y dibujar un diagrama lógico completo que incluya la numeración de pines.

47. El codificador que se muestra en el diagrama de bloques de la Figura 6.88 codifica cada pulsación de una tecla decimal y la convierte en BCD. Utilizar un 74HC147 para implementar esta función y dibujar un diagrama lógico completo que incluya la numeración de pines.

48. El sistema de la Figura 6.88 requiere dos convertidores de código. El convertidor BCD-binario convierte los dos números BCD de dos dígitos del registro A en un código binario de 8 bits (en realidad sólo se necesitan 7 bits dado que el bit más significativo siempre es 0). Utilizar los convertidores de código MSI apropiados para implementar la función del convertidor BCD-binario, y dibujar un diagrama lógico completo que incluya la numeración de pines.

Localización de fallos con el banco de trabajo electrónico

49. Abrir el archivo PR06_49.EWB y probar el circuito lógico para determinar si existe un fallo. Si existe, identificarlo si es posible.

50. Abrir el archivo PR06_50.EWB y probar el circuito lógico para determinar si existe un fallo. Si existe, identificarlo si es posible.

51. Abrir el archivo PR06_51.EWB y probar el circuito lógico para determinar si existe un fallo. Si existe, identificarlo si es posible.

52. Abrir el archivo PR06_52.EWB y probar el circuito lógico para determinar si existe un fallo. Si existe, identificarlo si es posible.

■ RESPUESTAS A LAS REVISIONES DE CADA SECCIÓN

SECCIÓN 6.1

1. (a) $\Sigma = 1, C_{out} = 0$ (b) $\Sigma = 0, C_{out} = 0$
   (c) $\Sigma = 1, C_{out} = 0$ (d) $\Sigma = 0, C_{out} = 1$
2. $\Sigma = 1, C_{out} = 1$

SECCIÓN 6.2

1. $C_{out}\Sigma_4\Sigma_3\Sigma_2\Sigma_1 = 11001$
2. Se requieren tres 74LS283 para sumar dos números de 10 bits.

SECCIÓN 6.3

1. A > B =1, A < B = 0, A = B = 0 cuando A = 1011 y B = 1010.
2. Comparador derecho: pin 7: A < B = 1; pin 6: A = B = 0; pin 5: A > B = 0.
   Comparador izquierdo: pin 7: A < B = 0; pin 6: A = B = 0; pin 5: A > B = 1.

SECCIÓN 6.4

1. La salida 5 está activa cuando en las entradas se aplica 101.
2. Se utilizan cuatro 74HC154 para decodificar un número binario de 6 bits.
3. La salida activa a nivel BAJO controla el display de diodos LED en cátodo común.

SECCIÓN 6.5

1. (a) $A_0 = 1, A_1 = 1, A_2 = 0, A_3 = 1$
   (b) No, no es un código BCD válido.
   (c) Sólo puede estar activada una entrada para obtener una salida válida.
2. (a) $\overline{A}_3 = 0, \overline{A}_2 = 1, \overline{A}_1 = 1, \overline{A}_0 = 1$
   (b) La salida es 0111, que es el complemento de 1000 (8).

SECCIÓN 6.6

1. 10000101 (BCD) = 1010101_2
2. Un convertidor binario-código Gray de ocho bits está formado por siete puertas OR-exclusiva en una disposición como la de la Figura 6.42.

SECCIÓN 6.7

1. La salida es 0.
2. (a) 74LS157: Selector cuádruple de dos entradas de datos.
   (b) 74LS151: Selector de datos de 8 entradas.
3. La salida de datos alterna entre un nivel BAJO y un nivel ALTO a medida que las entradas de selección de datos cambian, secuencialmente, entre los distintos estados binarios.
4. (a) El 74HC157A multiplexa los dos códigos BCD al decodificador de 7 segmentos.
   (b) El 74LS48 decodifica el código BCD para excitar el display.
   (c) El 74LS139 activa los displays de 7 segmentos alternativamente.

SECCIÓN 6.8

1. Se puede utilizar un decodificador como multiplexor, utilizando las líneas de entrada como entradas de selección de datos y una línea de activación como entrada de datos.
2. Las salidas están todas a nivel ALTO excepto D_{10}, que es un nivel BAJO.

SECCIÓN 6.9

1. (a) Paridad par: 1110100
   (b) Paridad par: 001100011
2. (a) Paridad impar: 11010101
   (b) Paridad impar: 11000001
3. (a) El código es correcto, cuatro 1s.
   (b) El código es erróneo, siete 1s.

SECCIÓN 6.10

1. Un glitch es un pico de tensión de muy corta duración (generalmente indeseado).
2. Los glitches los originan los estados de transición.
3. Validar (strobing) consiste en la activación de un dispositivo durante un periodo de tiempo especificado, mientras el dispositivo no se encuentra en un estado de transición

SECCIÓN 6.11

1. La secuencia de estados de entrada del decodificador es 00, 01, 11, 10.
2. Negativa-OR : $\overline{A} + \overline{B} = \overline{AB}$
3. El temporizador de larga duración se dispara cuando pasamos a los estados primero o tercero. El temporizador de corta duración se dispara cuando pasamos a los estados segundo o cuarto.

■ RESPUESTAS A LOS PROBLEMAS RELACIONADOS DE LOS EJEMPLOS

6.1 $\Sigma = 1, C_{out} = 1$

6.2 $\Sigma_1 = 0$
$\Sigma_2 = 0$
$\Sigma_3 = 1$
$\Sigma_4 = 1$

6.3 1011 + 1010 = 10101

6.4. Véase la Figura 6.89.

6.5. Véase la Figura 6.90.

6.6 $A > B = 0, A = B = 0, A < B = 1$

6.7. Véase la Figura 6.91.

6.8. Véase la Figura 6.92.

6.9. Salida 22

6.10. Véase la Figura 6.93.

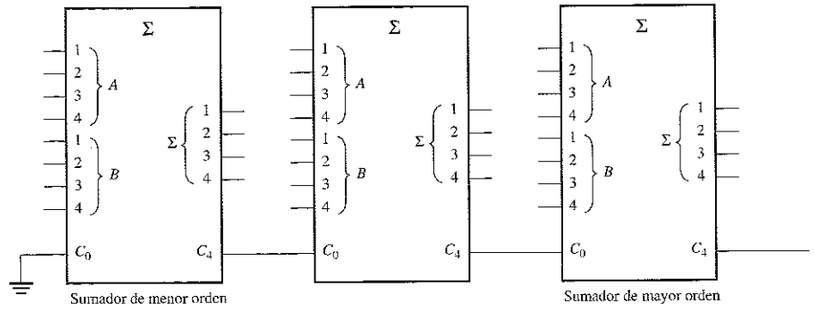


FIGURA 6.89

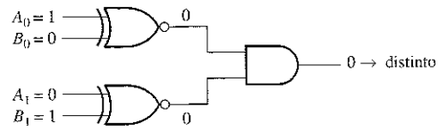


FIGURA 6.90

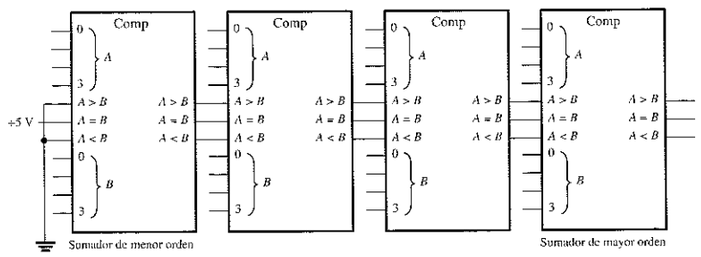


FIGURA 6.91

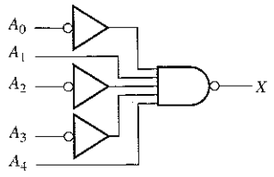


FIGURA 6.92

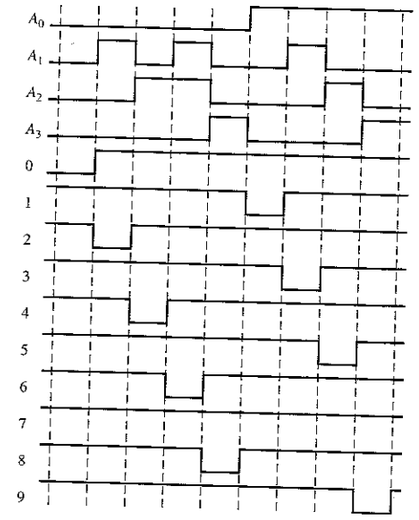


FIGURA 6.93

6.11. Todas las entradas a nivel BAJO: $\overline{A}_0 = 0, \overline{A}_1 = 1, \overline{A}_2 = 1, \overline{A}_3 = 0$
Todas las entradas a nivel ALTO: todas las salidas a nivel ALTO.

6.12 BCD 01000001

	00000001	1
	00101000	40
Binario	00101001	41

6.13 Siete puertas OR-exclusiva.

6.14. Véase la Figura 6.94.

6.15. D_0: $S_3 = 0, S_2 = 0, S_1 = 0, S_0 = 0$
D_4: $S_3 = 0, S_2 = 1, S_1 = 0, S_0 = 0$
D_8: $S_3 = 1, S_2 = 0, S_1 = 0, S_0 = 0$
D_{13}: $S_3 = 1, S_2 = 1, S_1 = 0, S_0 = 1$

6.16. Véase la Figura 6.95.

FIGURA 6.94

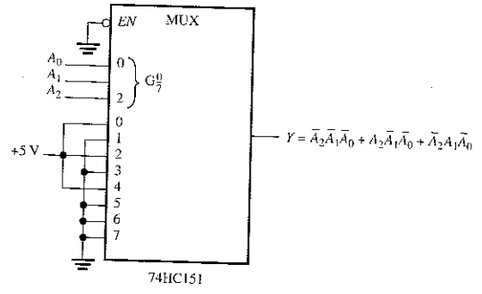


FIGURA 6.95

6.17. Véase la Figura 6.96.

6.18. Véase la Figura 6.97.

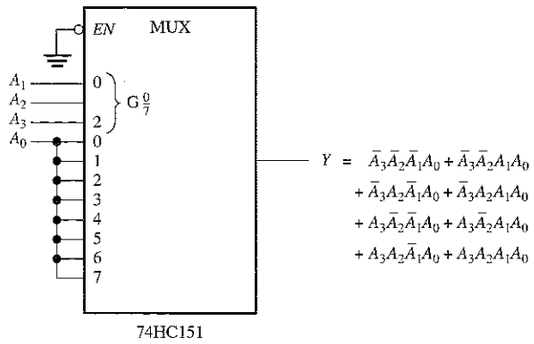


FIGURA 6.96

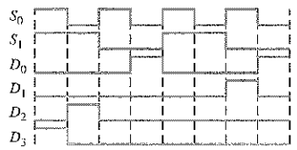


FIGURA 6.97

■ **RESPUESTAS AL AUTOTEST**

1. (a) **2.** (b) **3.** (c) **4.** (a) **5.** (d) **6.** (b) **7.** (c) **8.** (b)
9. (a) **10.** (d) **11.** (c) **12.** (f)

7

INTRODUCCIÓN A LOS DISPOSITIVOS LÓGICOS PROGRAMABLES

7.1 Dispositivos lógicos Programables (PLD). Clasificación
7.2 Matriz lógica programable (PAL)
7.3 Matriz lógica genérica (GAL)
7.4 La GAL22V10
7.5 La GAL16V8
7.6 Programación de los PLD
7.7 Software de los dispositivos PLD
7.8 Aplicación a los sistemas digitales

WWW Simulación por computadora de circuitos digitales:
El tutorial de Electronics Workbench (EWB) se encuentra en la dirección http://www.prenhall.com/floyd.

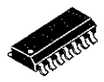
■ OBJETIVOS DEL CAPÍTULO

- ❑ Estudiar las matrices programables y describir la mayoría de los tipos de dispositivos lógicos programables.
- ❑ Describir la estructura básica de una PAL y explicar cómo, programando la matriz AND, se generan las funciones lógicas estándar.
- ❑ Describir la estructura básica de una GAL y explicar cómo programando la matriz AND se generan funciones lógicas estándar.
- ❑ Describir la arquitectura básica de la GAL22V10.
- ❑ Describir la arquitectura básica de la GAL16V8.
- ❑ Estudiar el procedimiento básico de programación de los PLD.
- ❑ Escribir sencillos programas en ABEL para implementar lógica combinacional en un PLD.
- ❑ Implementar lógica combinacional sencilla utilizando una GAL.

■ INTRODUCCIÓN

En este capítulo se introducen los dispositivos lógicos programables (PLD, Programmable Logic Device). Se ha puesto un especial énfasis en la utilización de los PLD para implementar lógica combinacional. En el Capítulo 11, se tratan las aplicaciones lógicas secuenciales de los PLD. En muchas aplicaciones, los PLD y, en concreto, las matrices lógicas programables (PAL, Programmable Array Logic) y las matrices lógicas genéricas (GAL, Generic Array Logic) pueden emplearse para reemplazar dispositivos lógicos SSI y MSI, consiguiendo con ello una reducción de etapas y de los costes.

Para implementar diseños lógicos con PLD se utilizan paquetes software de programación, tales como ABEL y CUPL. En este libro, se utiliza ABEL para ilustrar los principios básicos de programación. Una vez que se haya familiarizado con un lenguaje de programación para PLD, los demás pueden aprenderse fácilmente, ya que son muy similares. El estudio que se hace de ABEL en este texto es limitado y no pretende ser un tratamiento exhaustivo, por lo que se recomienda el uso de la documentación del fabricante y de otras fuentes para complementar esta exposición. El estudio de este capítulo se puede posponer hasta después de ver el Capítulo 10, o se puede omitir por completo sin que afecte a otros temas, excepto a los tratados en el Capítulo 11.

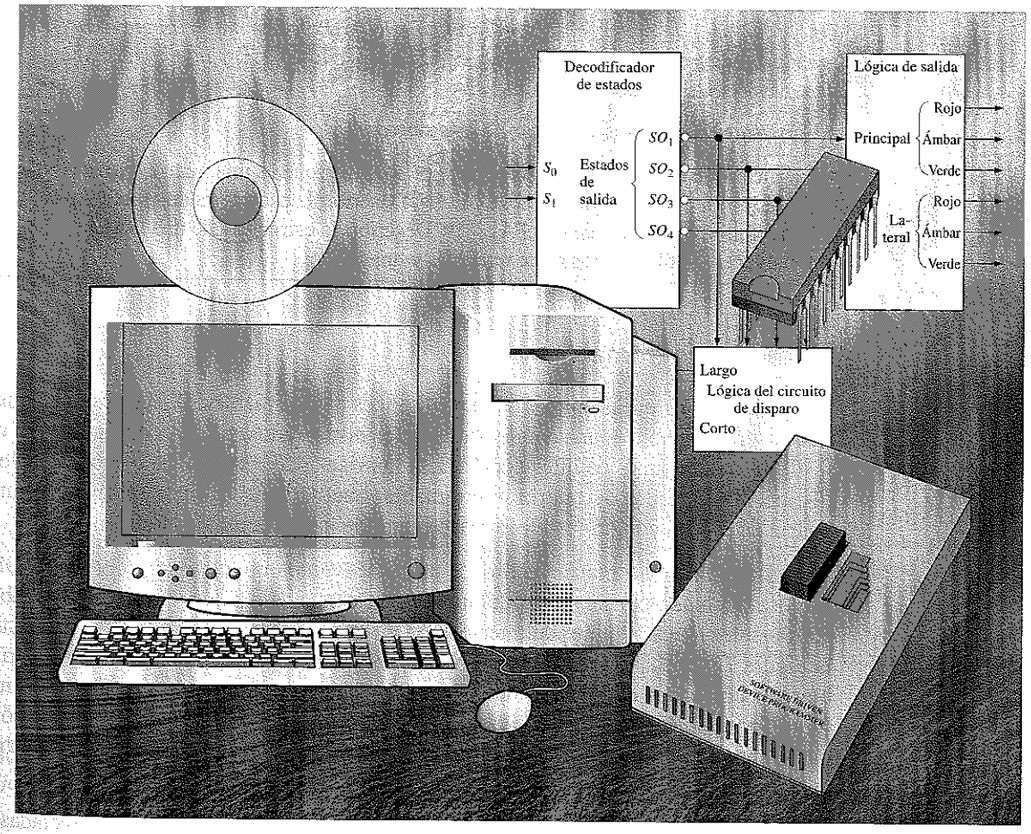


DISPOSITIVOS ESPECÍFICOS

PAL16L8 GAL22V10 GAL16V8
GAL20V8

■ APLICACIÓN A UN SISTEMA DIGITAL

Este sistema digital ilustra los conceptos que se estudian en el capítulo. Se rediseña la parte de la lógica combinacional del sistema de control de semáforos del Capítulo 6, utilizando un dispositivo programable. Se desarrolla un programa en lenguaje ABEL, para implementar la decodificación de estados y la lógica de salida.



7.1 ■ DISPOSITIVOS LÓGICOS PROGRAMABLES (PLD). CLASIFICACIÓN

Los dispositivos lógicos programables (PLD) se utilizan en muchas aplicaciones para reemplazar a los circuitos SSI y MSI, ya que ahorran espacio y reducen el número y el coste de los dispositivos en un determinado diseño. Un PLD está formado por una matriz de puertas AND y puertas OR, que se puede programar para conseguir funciones lógicas específicas. Existen cuatro tipos de dispositivos que se clasifican como PLD: la memoria programable de sólo lectura (PROM, Programmable Read-Only Memory), la matriz lógica programable PLA (Programmable Logic Array), la matriz lógica programable PAL (Programmable Array Logic) y la matriz lógica genérica (GAL, Generic Array Logic).

Al finalizar esta sección, el lector deberá ser capaz de:

- Explicar qué es una matriz básica OR.
- Explicar qué es una matriz básica AND.
- Describir la estructura de una PROM.
- Describir la estructura de una PLA.
- Describir la estructura de una PAL.
- Describir la estructura de una GAL.
- Establecer las diferencias entre una PAL y una GAL.

NOTAS INFORMÁTICAS
La mayor parte de los diseños de nivel de sistema incorporan diversos dispositivos, como son las memorias RAM, ROM, controladores, procesadores, etc., que se interconectan mediante gran cantidad de dispositivos lógicos de propósito general, frecuentemente denominados lógica de "unión" (glue logic). En los últimos años, los dispositivos PLD han comenzado a reemplazar muchos de los antiguos dispositivos de "unión" SSI y MSI. El uso de dispositivos PLD proporciona una reducción en el número de circuitos integrados. Por ejemplo, en los sistemas de memoria de las computadoras, los PLD pueden utilizarse para decodificar direcciones de memoria y generar señales de escritura en memoria.

Matrices programables

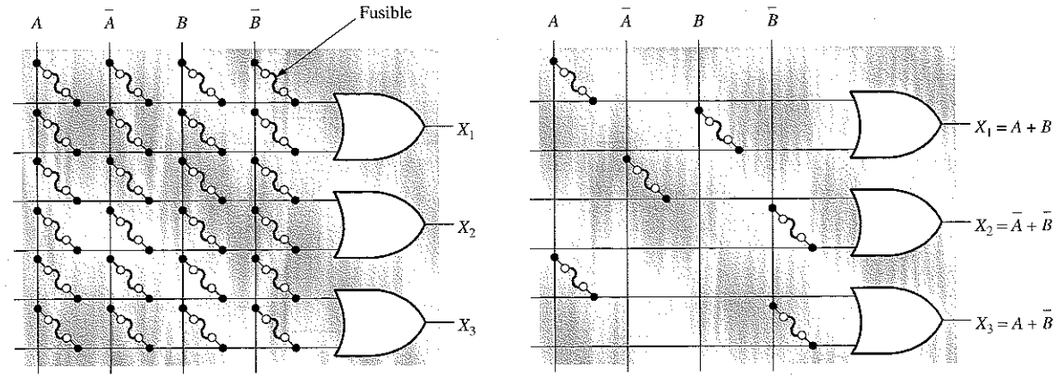
Todos los PLD están formados por matrices programables. Esencialmente, una **matriz** programable es una red de conductores distribuidos en filas y columnas con un fusible en cada punto de intersección. Las matrices pueden ser fijas o programables. El tipo más sencillo de matriz programable, que data de los años sesenta, era una matriz de diodos con un fusible en cada punto de intersección de la misma.

La matriz OR. La matriz de diodos primera evolucionó a la matriz integrada OR, que está formada por una serie de puertas OR conectadas a una matriz programable con fusibles en cada punto de intersección de una columna y una fila, como muestra la Figura 7.1(a). La matriz se programa fundiendo los **fusibles** para eliminar las variables seleccionadas de las funciones de salida, como ilustra la parte (b) de la misma figura, para un caso específico. Para cada una de las entradas de una puerta OR, sólo queda intacto un fusible que conecta la variable deseada a la entrada de la puerta. Una vez que el fusible está fundido, no se puede volver a conectar.

La matriz AND. Este tipo de matriz está formado por puertas AND conectadas a una matriz programable con fusibles en cada punto de intersección, como muestra la Figura 7.2(a). Al igual que la matriz OR, la matriz AND se programa fundiendo los fusibles para eliminar las variables de la función de salida, como ilustra la parte (b) de la figura. Para cada entrada de una puerta AND, sólo queda intacto un fusible que conecta la variable deseada a la entrada de la puerta. Como para la matriz OR, la matriz AND con fusibles se puede programar una única vez.

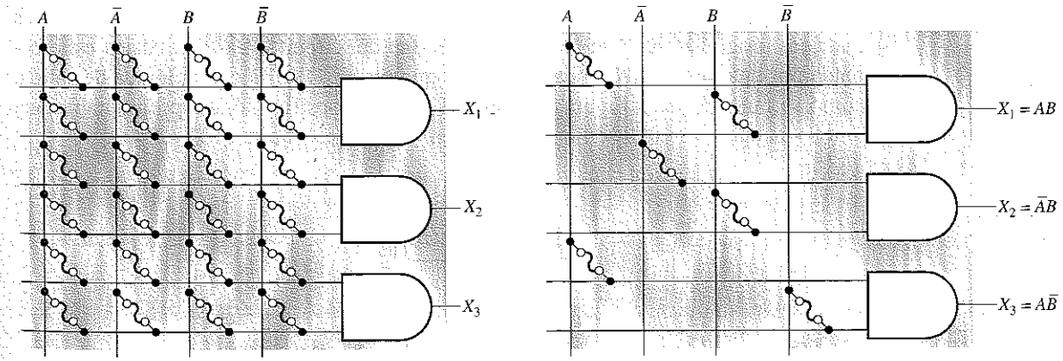
Clasificación de los PLD

Los PLD se clasifican de acuerdo con su **arquitectura**, la cual es básicamente la ordenación funcional de los elementos internos que proporciona al dispositivo sus características de operación específicas.



(a) No programada (b) Programada

FIGURA 7.1



(a) No programada (b) Programada

FIGURA 7.2
Ejemplo de una matriz básica AND.

Memoria programable de sólo lectura (PROM, Programmable Read-Only Memory). La **PROM** está formada por un conjunto fijo (no programable) de puertas AND conectadas como decodificador y una matriz programable OR, como muestra el diagrama de bloques general de la Figura 7.3. Fundamentalmente, la PROM se utiliza como una memoria direccionable y no como un dispositivo lógico, debido a las limitaciones que imponen las puertas AND fijas. En el Capítulo 12 se cubren en detalle estas memorias.

Matriz lógica programable PLA (Programmable Logic Array). La **PLA** es un PLD formado por una matriz AND programable y una matriz OR programable, como muestra la Figura 7.4. La PLA ha sido desarrollada para superar algunas de las limitaciones de las memorias PROM. La PLA también se denomina FPLA (Field-pro-

grammable logic array, matriz lógica programable en campo) debido a que es el usuario y no el fabricante el que la programa.

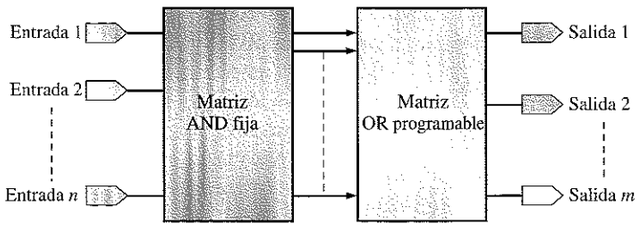


FIGURA 7.3
Diagrama de bloques de una PROM (Programmable read-only memory).

Matriz lógica programable PAL (Programmable Array Logic). La **PAL** es un PLD que se ha desarrollado para superar ciertas desventajas de la PLA, tales como los largos retardos debidos a los fusibles adicionales que resultan de la utilización de dos matrices programables y la mayor complejidad del circuito. La PAL básica está formada por una matriz AND programable y una matriz OR fija con la lógica de salida, como se muestra en la Figura 7.5. La PAL es el dispositivo lógico programable una única vez más común, y se implementa con tecnología bipolar (TTL o ECL).

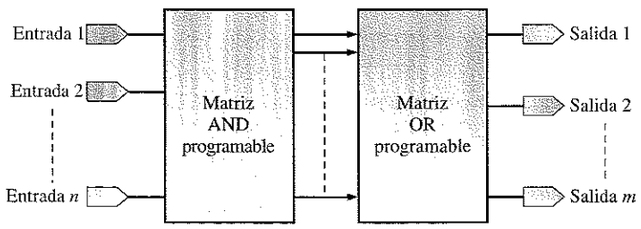


FIGURA 7.4
Diagrama de bloques de una PLA (Programmable logic array).

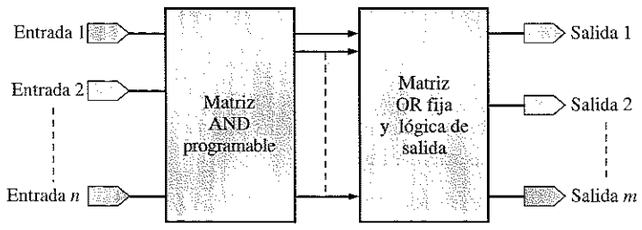


FIGURA 7.5
Diagrama de bloques de una PAL (Programmable array logic).

Matriz lógica genérica (GAL, Generic Array Logic). El desarrollo más reciente de los PLD es la GAL. La **GAL**, al igual que la PAL, se forma con una matriz AND programable y una matriz OR fija, con una salida lógica programable. Las dos principales diferencias entre los dispositivos GAL y PAL son: (a) la GAL es reprogramable y (b) la GAL tiene configuraciones de salida programables.

La GAL se puede programar una y otra vez, ya que usa la tecnología **E²CMOS** (Electrically Erasable CMOS, CMOS borrable eléctricamente), en lugar de tecnología bipolar y fusibles. En la Figura 7.6 se muestra el diagrama de bloques de una GAL.

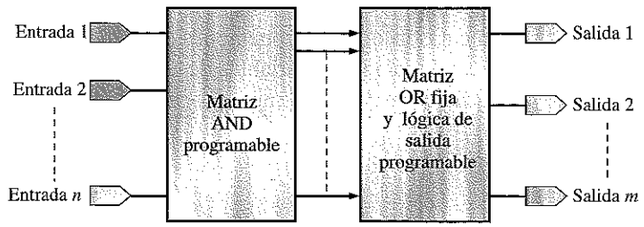


FIGURA 7.6
Diagrama de bloques de una GAL (Generic array logic).

REVISIÓN DE LA SECCIÓN 7.1

1. Enumerar cuatro dispositivos lógicos programables (PLD).
2. ¿En qué se diferencian una PLA y una PAL?
3. Describir las diferencias entre una PAL y una GAL.

7.2 ■ MATRIZ LÓGICA PROGRAMABLE (PAL, PROGRAMMABLE ARRAY LOGIC)

Los dispositivos lógicos programables que se usan más comúnmente para la implementación lógica son la PAL y la GAL. PAL es una denominación que, originalmente, utilizaba la compañía Monolithic Memories, Inc. (ahora forma parte de Advanced Micro Devices, AMD) y, más tarde, otros fabricantes obtuvieron también la licencia. Como ya vimos en la sección anterior, la PAL, en su forma básica, es un PLD formado por una matriz AND programable una única vez y una matriz OR fija. En esta sección, aprenderemos a utilizar las PAL para generar funciones lógicas combinacionales específicas y examinaremos una PAL concreta.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir el funcionamiento básico de una PAL. ■ Mostrar cómo se implementa una suma de productos con una PAL. ■ Entender los diagramas lógicos simplificados de una PAL. ■ Explicar los tres tipos básicos de lógica combinacional de salida para una PAL. ■ Interpretar las referencias estándar de las PAL. ■ Utilizar la PAL16L8.

Funcionamiento de la PAL

Una PAL está formada por una matriz de puertas AND programable conectada a una matriz de puertas OR fija. Esta estructura permite implementar cualquier suma

de productos lógica con un número de variables definido. Recuérdese que, en el Capítulo 4, vimos que cualquier función lógica puede expresarse como suma de productos.

En la Figura 7.7 se ilustra la estructura básica de una PAL para dos variables de entrada y una salida, aunque la mayoría de las PAL tiene muchas entradas y muchas salidas. Como se sabe, una matriz programable es esencialmente una red de conductores ordenados en filas y columnas, con un fusible en cada punto de intersección. Cada punto de intersección entre una fila y una columna se denomina **celda** y es el elemento programable de la PAL. Cada fila se conecta a la entrada de una puerta AND y cada columna a una variable de entrada o a su complemento. En función de la presencia o ausencia de las conexiones creadas por programación, se puede aplicar cualquier combinación de variables de entrada o sus complementos a una puerta AND, para generar cualquier término producto que se desee.

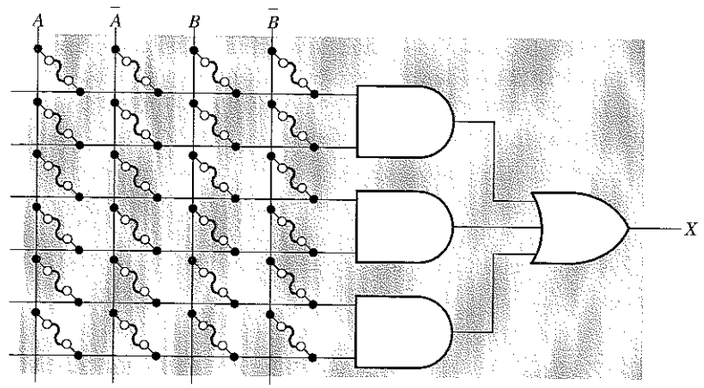


FIGURA 7.7
Estructura básica de una PAL.

Implementación de una suma de productos. En su forma más sencilla, cada celda de una matriz AND básica está formada por un fusible que conecta una fila y una columna, como se representa en la Figura 7.7. Cuando se requiere la conexión entre una fila y una columna, el fusible queda intacto. Cuando dicha conexión no se requiere, el fusible se abre en el proceso de programación.

En la Figura 7.8 se muestra un ejemplo de una matriz programada para generar el producto AB en la salida de la puerta AND superior, el producto $A\bar{B}$ en la puerta AND del centro y el producto $\bar{A}\bar{B}$ en la puerta AND inferior. Como puede verse, los fusibles quedan intactos para conectar las variables deseadas, o sus complementos, a las entradas apropiadas de las puertas AND. Los fusibles se funden en los casos en que las variables o sus complementos no se utilizan en el producto que se desea obtener. La salida de la puerta OR proporciona la suma de productos.

$$X = AB + A\bar{B} + \bar{A}\bar{B}$$

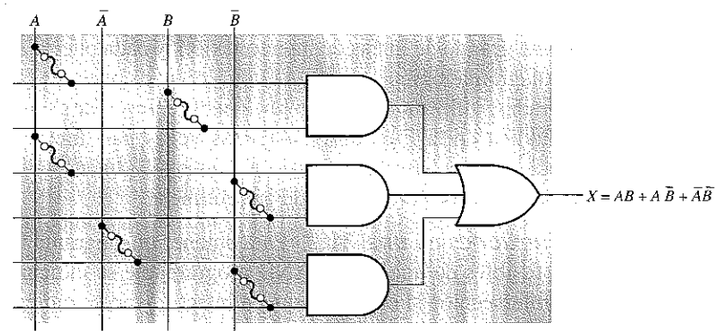


FIGURA 7.8
Implementación mediante una PAL de una suma de productos.

Símbolos simplificados

Vamos a ver cómo se representa un pequeño segmento de una PAL típica. Las PAL reales tienen muchas puertas AND y muchas puertas OR, además de otra circuitería, y son capaces de manejar muchas variables de entrada y sus complementos. Puesto que las PAL son circuitos integrados muy complejos, los fabricantes han adoptado una notación simplificada para eliminar los diagramas lógicos arrolladoramente complicados.

Buffers de entrada. Para evitar cargar con la gran cantidad de entradas de puertas AND a las que se pueden conectar una variable o su complemento, se añade un buffer a las variables de entrada de la PAL. Un buffer inversor genera el complemento de una variable de entrada. El símbolo que representa el circuito **buffer**, el cual genera la variable y su complemento en sus salidas, se muestra en la Figura 7.9, donde la salida con el círculo corresponde a la variable complementada.

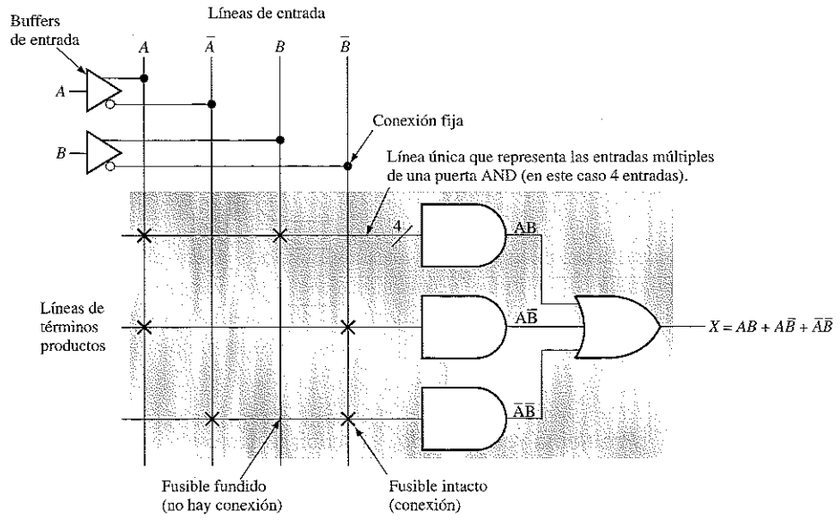


FIGURA 7.9
Diagrama simplificado de una PAL programada.

Puertas AND. Una matriz AND de una PAL típica tiene una cantidad extremadamente grande de líneas de interconexión, y cada puerta AND tiene entradas múltiples. El diagrama lógico de una PAL muestra cada puerta AND, que realmente tiene varias entradas, utilizando una sola línea para representar a todas las líneas de entrada, como se indica en la Figura 7.9. Algunas veces, las líneas múltiples también se indican mediante una barra inclinada y el número de líneas, tal como se muestra en la puerta AND superior para el caso de cuatro líneas.

Conexiones en una PAL. Para obtener un diagrama lógico lo más sencillo posible, los fusibles de una matriz AND programable se indican mediante una X en el punto de intersección si el fusible queda intacto, y no se indica nada si el fusible está fundido, como se indica en la Figura 7.9. Las conexiones fijas emplean el punto estándar, tal y como se muestra.

EJEMPLO 7.1 Indicar cómo se programa una PAL para obtener la siguiente función lógica de tres variables:

$$X = A\overline{B}C + \overline{A}B\overline{C} + \overline{A}\,\overline{B} + AC$$

Solución. En la Figura 7.10 se muestra la matriz programada. Los fusibles intactos se indican con una X. La ausencia de X significa que los fusibles están fundidos.

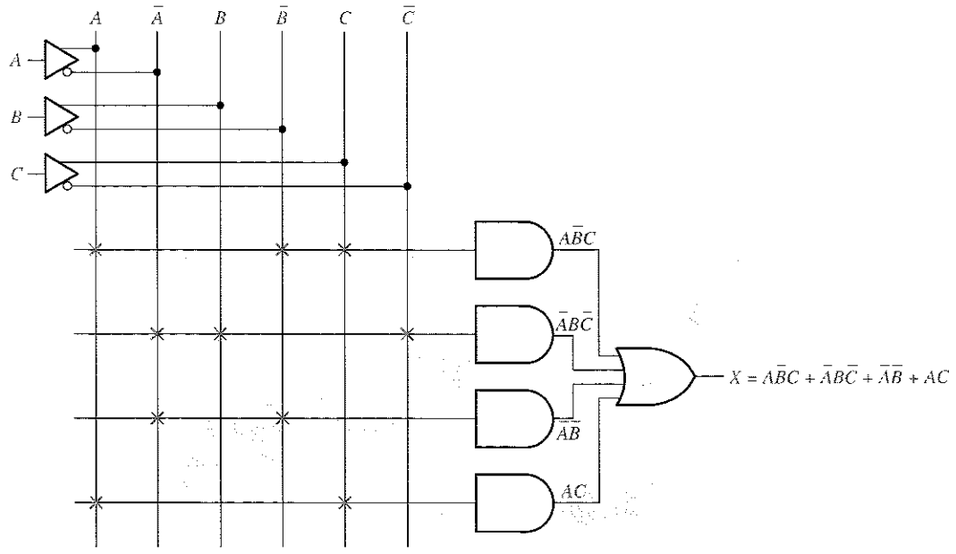


FIGURA 7.10

Problema relacionado. Escribir la expresión de salida si los fusibles que conectan la entrada *A* con la fila superior e inferior de la Figura 7.10 están fundidos.

Diagrama de bloques de una PAL

En la Figura 7.11 se muestra el diagrama de bloques de una PAL. Las salidas de la matriz AND son las entradas de la matriz OR, y la salida de cada puerta OR se asocia a una lógica de salida. Una PAL típica tiene ocho o más entradas en la matriz AND y hasta ocho bloques lógicos de salida como se indica en la figura, siendo $n \geq 8$ y $m \leq 8$. Algunas PAL disponen de pines de entrada/salida (E/S) combinados, que se pueden programar como salida o como entrada. El símbolo ⇔ significa que un pin puede utilizarse como entrada o como salida.

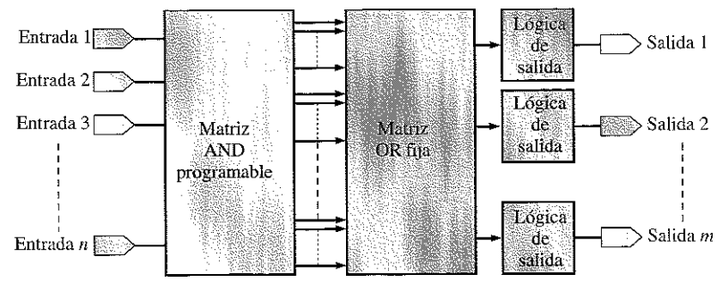


FIGURA 7.11
Diagrama de bloques de una PAL.

Lógica combinacional de salida de una PAL

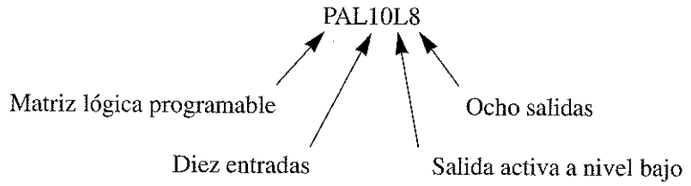
Existen varios tipos básicos de circuitos lógicos de salida para una PAL, que permiten configurar el dispositivo para una aplicación específica. En este capítulo, sólo se van a tratar los aspectos de la lógica de salida relativos a las funciones lógicas combinacionales. En el Capítulo 11 se cubren otras funciones adicionales, después de haber estudiado los flip-flops, contadores y registros. En la Figura 7.12 se muestran tres tipos básicos de lógica combinacional de salida con salidas de tres estados y la puerta OR asociada. Los tipos de lógica de salida de una PAL son:

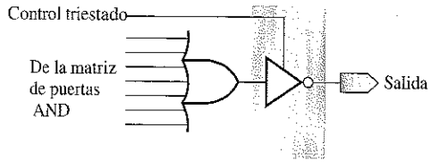
- *Salida combinacional.* Esta salida se usa para una suma de productos y, usualmente, la salida puede ser activa a nivel ALTO o BAJO.
- *Entrada/salida combinacional.* Esta salida se usa cuando la función de salida debe realimentarse a una entrada de la matriz, o cuando el pin de E/S se usa sólo como entrada.
- *Salida de polaridad programable.* Esta salida se usa para seleccionar la función de salida o su complemento, programando la puerta OR-exclusiva para realizar la inversión o no. El fusible de la entrada de la puerta OR-exclusiva se funde en el caso de inversión, y queda intacto en el caso de no inversión.

Referencia estándar de las PAL

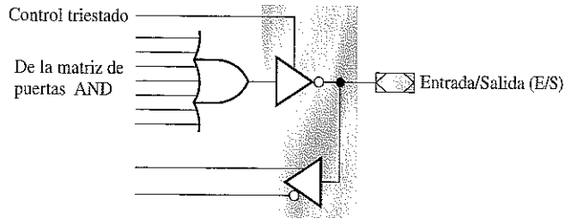
Existe una gran variedad de configuraciones de PAL estándar, cada una de las cuales se identifica mediante una única referencia. Esta referencia siempre comienza con el

prefijo PAL. Los dos primeros dígitos que siguen al prefijo indican el número de entradas, en el que se incluye el número de salidas que se pueden configurar como entradas. La letra que sigue al número de entradas designa el tipo de salida: L = active-LOW (activa a nivel BAJO), H = active-HIGH (activa a nivel ALTO) y P = polaridad programable. El dígito o los dos dígitos que siguen a la letra que designa el tipo de salida indican el número de salidas. Veamos un ejemplo:

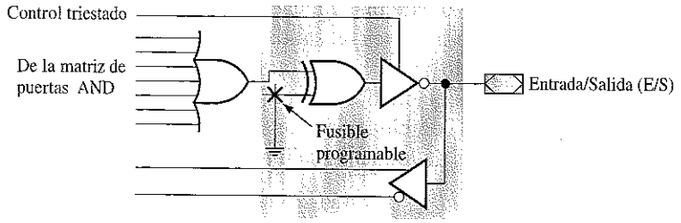




(a) Salida combinacional (activa a nivel BAJO). Una salida activa a nivel ALTO no tendría el círculo en el símbolo de la puerta triestado.



(b) Entrada/Salida combinacional (activa a nivel BAJO).



(c) Salida de polaridad programable.

FIGURA 7.12
Tipos básicos de la lógica de salida combinacional de una PAL.

Además, una referencia de una PAL puede llevar sufijos que especifican la velocidad, el tipo de encapsulado y el rango de temperatura.

En la Figura 7.13 se muestra, como ejemplo de configuración de una PAL, el diagrama de bloques del dispositivo PAL16L8. Este dispositivo tiene 10 entradas dedicadas, dos salidas dedicadas y 6 pines que se pueden utilizar como entradas o como salidas. Las salidas son activas a nivel BAJO.

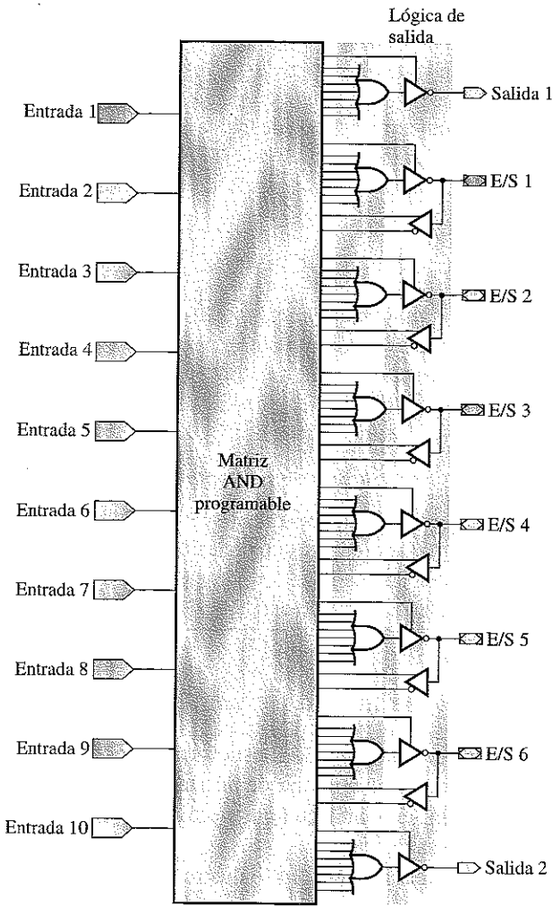


FIGURA 7.13
Diagrama de bloques de la PAL16L8.

EJEMPLO 7.2 Determinar el número de entradas, el número de salidas y el tipo de salida de cada una de las siguientes referencias de dispositivos PAL:

(a) PAL12H6 **(b)** PAL16L2 **(c)** PAL20P8

Solución

(a) 12 entradas, 6 salidas, salidas activas a nivel ALTO.
(b) 16 entradas, 2 salidas, salidas activas a nivel BAJO.
(c) 20 entradas, 8 salidas, salidas de polaridad programable.

Problema relacionado. Describir la PAL cuya referencia es PAL14H4.

REVISIÓN DE LA SECCIÓN 7.2

1. ¿Qué es una PAL?
2. Explicar el propósito de un fusible en una PAL.
3. ¿Puede implementarse cualquier función lógica con una PAL?
4. Enumerar tres tipos de lógica de salida de una PAL.
5. ¿Qué significa la referencia PAL12H6?

7.3 ■ MATRIZ LÓGICA GENÉRICA (GAL, GENERIC ARRAY LOGIC)

GAL es una denominación que utilizaba originalmente Lattice Semiconductor y que más tarde se licenció a otros fabricantes. Como hemos visto en la Sección 7.1, la GAL en su forma básica es un PLD con una matriz AND reprogramable, una matriz OR fija y una salida lógica programable. En esta sección, se introducen los conceptos básicos, y en las secciones 7.4 y 7.5 se van a examinar GAL específicas.

Al finalizar esta sección, el lector deberá ser capaz de.

■ Describir el funcionamiento básico de una GAL. ■ Demostrar cómo se implementa una suma de productos con una GAL. ■ Comparar la celda E²CMOS de una GAL con un fusible de una PAL. ■ Definir OLMC y explicar su propósito. ■ Interpretar la referencia estándar de una GAL.

Funcionamiento de una GAL

Básicamente, una GAL está formada por una matriz de puertas AND reprogramable conectada a una matriz de puertas OR fija. Al igual que una PAL, esta estructura permite implementar cualquier expresión lógica suma de productos con un número de variables definido.

En la Figura 7.14 se ilustra la estructura básica de una GAL con dos variables de entrada y una salida, aunque la mayoría de las GAL pueden tener muchas entradas y muchas salidas. La matriz reprogramable es esencialmente una red de conductores ordenados en filas y columnas, con una celda CMOS eléctricamente borrable (E²CMOS) en cada punto de intersección, en lugar de un fusible como en el caso de las PAL. En la figura, estas celdas se indican como bloques.

Cada fila está conectada a la entrada de una puerta AND, y cada columna a una variable de entrada o a su complemento. Mediante la programación se *activa* o *desactiva* cada celda E²CMOS, y se puede aplicar cualquier combinación de variables de entrada, o sus complementos, a una puerta AND para generar cualquier término producto que se desee. Una celda *activada* conecta de forma efectiva su correspondiente fila y columna, y una celda *desactivada* desconecta la fila y la columna. Las celdas se pueden borrar y reprogramar eléctricamente. Una celda E²CMOS típica puede mantener el estado en que se ha programado durante 20 años o más. Puede consultarse en el Capítulo 15 una exposición sobre la forma en que trabaja una celda E²CMOS.

Implementación de una suma de productos. En la Figura 7.15 se muestra un ejemplo de una sencilla matriz GAL, programada para obtener el producto $\overline{A}B$ en la puerta AND superior, AB en la puerta del centro y $\overline{A}\overline{B}$ en la puerta AND inferior.

Como se indica, las celdas E²CMOS *activadas* conectan las variables deseadas o sus complementos con las apropiadas entradas de las puertas AND. Las celdas E²CMOS están *desactivadas* cuando una variable o su complemento no se utiliza en un determinado producto. La salida final de la puerta OR es una suma de productos.

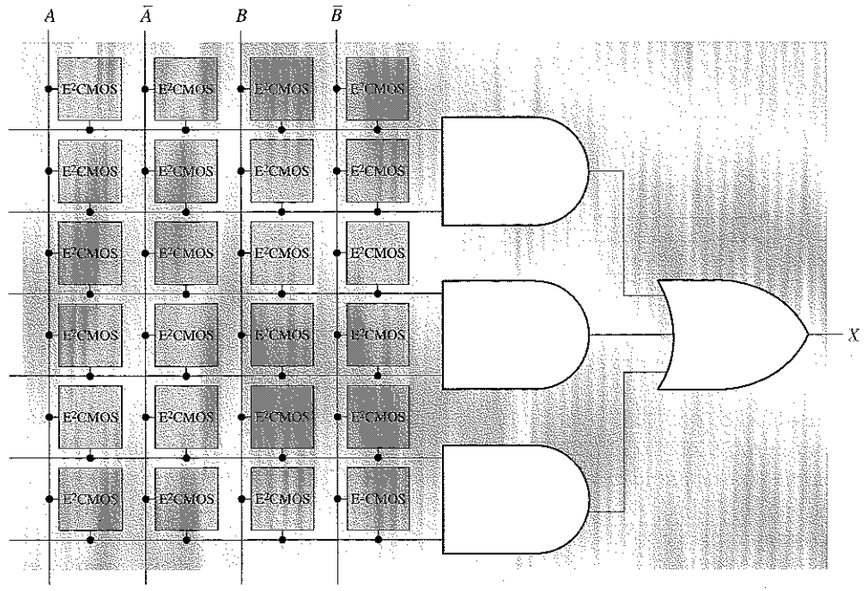


FIGURA 7.14
Estructura básica de una matriz E²CMOS de una GAL.

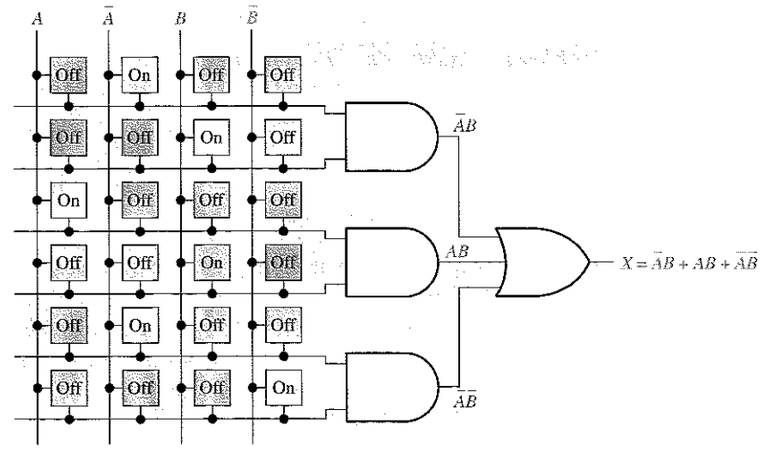


FIGURA 7.15
Implementación de una suma de productos con una GAL.

EJEMPLO 7.3 Indicar cómo se programa una GAL para obtener la siguiente expresión de 3 variables:

$$X = \bar{A}B\bar{C} + \bar{A}BC + BC + A\bar{B}$$

Solución

En la Figura 7.16 se muestra la matriz programada en notación simplificada. Las celdas que están *activadas* se marcan con una X. La ausencia de X significa que la celda está *desactivada.*

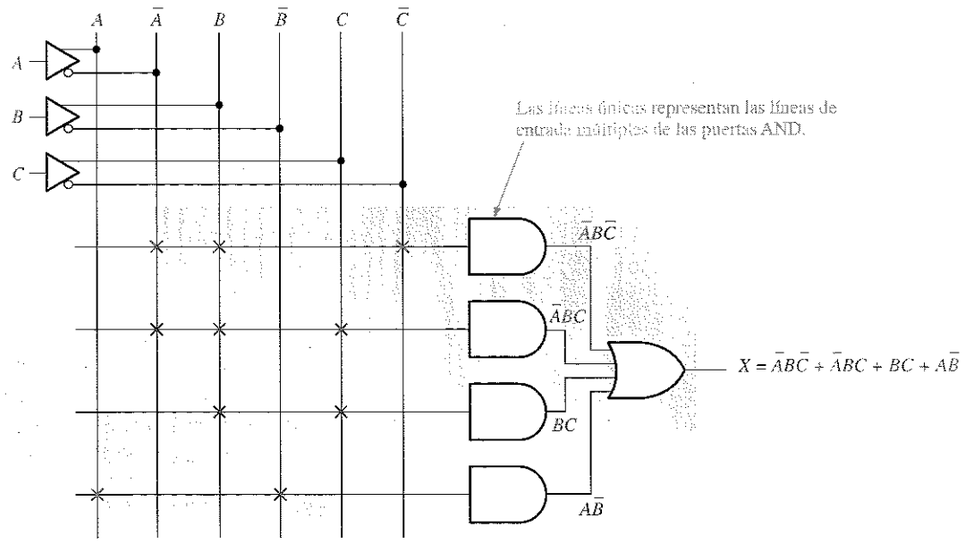


FIGURA 7.16

Problema relacionado. Escribir la expresión de salida si las celdas que conectan la entrada con la primera y la segunda filas de la Figura 7.16 están *desactivadas.*

Diagrama de bloques de una GAL

En la Figura 7.17 se presenta el diagrama de bloques de una GAL. Las salidas de la matriz AND se introducen en las macroceldas lógicas de salida (**OLMC**, output logic macrocells), que contienen puertas OR y lógica programable. Una GAL típica puede tener ocho o más entradas a su matriz AND, y ocho o más entradas/salidas de las OLMC como se indica en la figura, siendo $n \geq 8$ y $m \geq 8$.

Las macroceldas lógicas de salida (OLMC) están formadas por circuitos lógicos que se pueden programar como *lógica combinacional* o como *lógica secuencial.* Las OLMC proporcionan mucha más flexibilidad que la lógica de salida fija de una PAL. En este capítulo, nos vamos a ocupar de las configuraciones de salida combinacionales, y en el Capítulo 11 se verán las configuraciones secuenciales, después de que se hayan estudiado los flip-flops, contadores y registros.

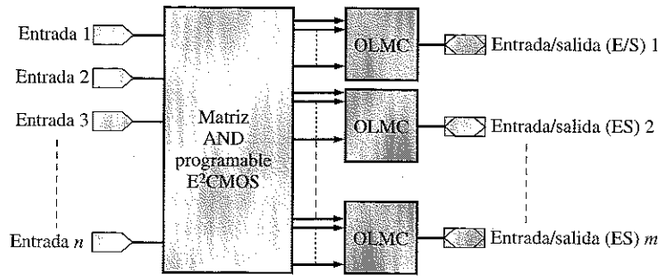
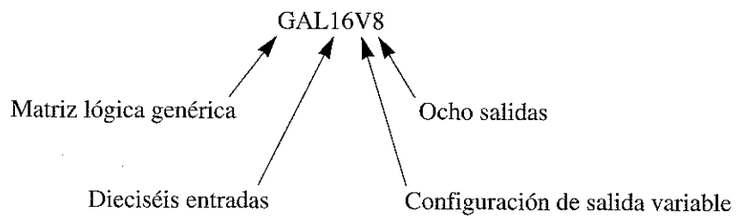


FIGURA 7.17
Diagrama de bloques de una GAL.

Referencia estándar de una GAL

Las GAL existen en una gran variedad de configuraciones, cada una de las cuales se identifica por una única referencia. Esta referencia siempre comienza por el prefijo GAL. Los dos primeros dígitos que siguen al prefijo indican el número de entradas, incluyendo las salidas que se pueden configurar como entradas. La letra V que sigue al número de entradas designa una configuración de salida variable. El número o los dos números siguientes al tipo de salida especifican el número de salidas. La siguiente referencia es un ejemplo.



EJEMPLO 7.4 Determinar el número de entradas y el número de salidas de cada una de las siguientes referencias.

(*a*) GAL20V8 (**b**) GAL22V10

Solución

(*a*) 20 entradas, 8 salidas.
(*b*) 22 entradas, 10 salidas.

Problema relacionado. Describir una GAL cuya referencia es GAL18V10.

REVISIÓN DE LA SECCIÓN 7.3

1. ¿Qué es una GAL?
2. Explicar el propósito de la celda E²CMOS en una GAL.
3. ¿Se puede implementar cualquier función lógica combinacional con una GAL?
4. ¿Qué es la OLMC?

7.4 ■ LA GAL22V10

Las diversas GAL tienen el mismo tipo de matriz programable. Se diferencian en el tamaño de la matriz, en el tipo de OLMC y en los parámetros de funcionamiento, tales como velocidad y disipación de potencia. Esta sección se ocupa de la popular matriz lógica genérica GAL22V10, con el fin de ilustrar los principios comunes a todos los tipos de GAL. En la siguiente sección veremos la GAL16V8.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar el funcionamiento de la OLMC y enumerar las cuatro configuraciones. ■ Describir el modo combinacional. ■ Establecer el funcionamiento y propósito del buffer de tres estados. ■ Explicar cómo la OLMC puede operar como entrada o como salida. ■ Describir cómo están organizadas las celdas E²CMOS. ■ Ilustrar cómo se implementa una suma de productos.

Diagrama de bloques de la GAL22V10

Como muestra el diagrama de bloques de la Figura 7.18, la GAL22V10 tiene doce entradas dedicadas y diez entradas/salidas (E/S). Este dispositivo se puede encontrar en encapsulado DIP (dual in-line) de 24 pines o PLCC (plastic chip carrier) de 28 pines, como muestra la Figura 7.19. Este dispositivo también está disponible en una versión de baja tensión: GAL22LV10.

Las macroceldas lógicas de salida (OLMC)

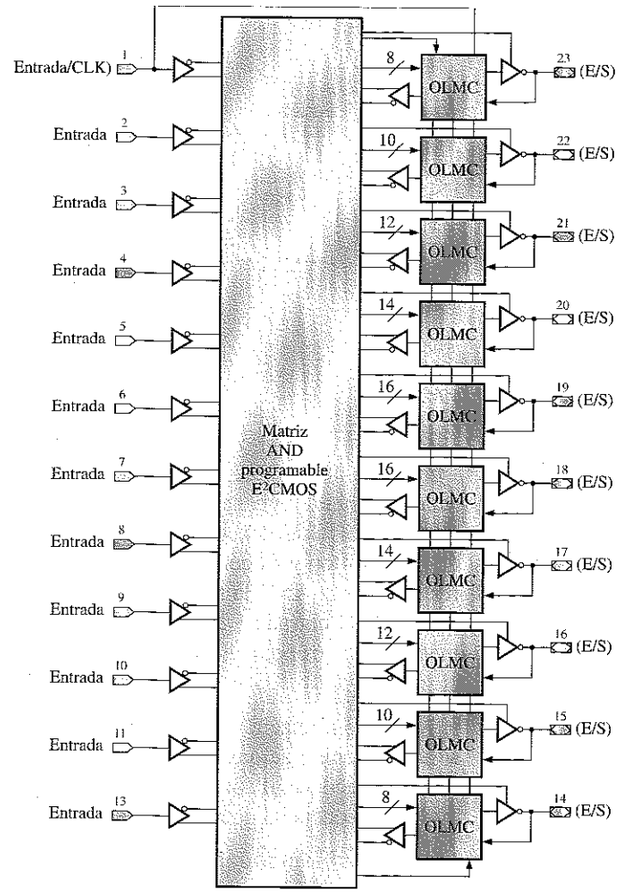
Como se ha dicho anteriormente, una OLMC contiene circuitos lógicos programables que se pueden configurar como entrada o salida combinacional, o como salida secuencial. En el modo secuencial (que se verá en el Capítulo 11) la salida procede de un flip-flop. Este capítulo se centra en el modo combinacional. Las configuraciones combinacionales de estas macroceldas lógicas se establecen automáticamente mediante programación.

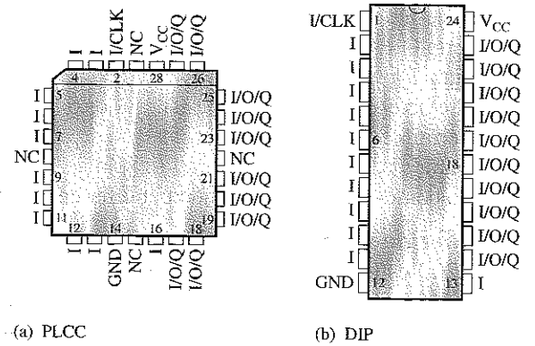
Como se indica en el diagrama de bloques de la Figura 7.18, de las diez OLMC disponibles de la GAL22V10, dos tienen ocho términos producto (número de líneas de la matriz AND a la puerta OR), dos tienen diez términos producto, otras dos doce, otras dos catorce y, por último, otras dos tienen dieciséis términos producto. Cada OLMC puede programarse para que el nivel activo de salida sea ALTO o BAJO y también, pueden programarse como entradas.

Diagrama lógico. En la Figura 7.20 se muestra un diagrama lógico básico de la OLMC de la GAL22V10. Las entradas de las puertas AND a la puerta OR varían des-

de diez hasta dieciséis, como se indica. La lógica de la zona sombreada está formada por un flip-flop (definido en el Capítulo 1 y que se estudia en el Capítulo 8, pero no se usa en este capítulo) y dos multiplexores (estudiados en el Capítulo 6).

FIGURA 7.18
Diagrama de bloques de la GAL22V10.





(a) PLCC (b) DIP

FIGURA 7.19
Encapsulados de la GAL22V10.

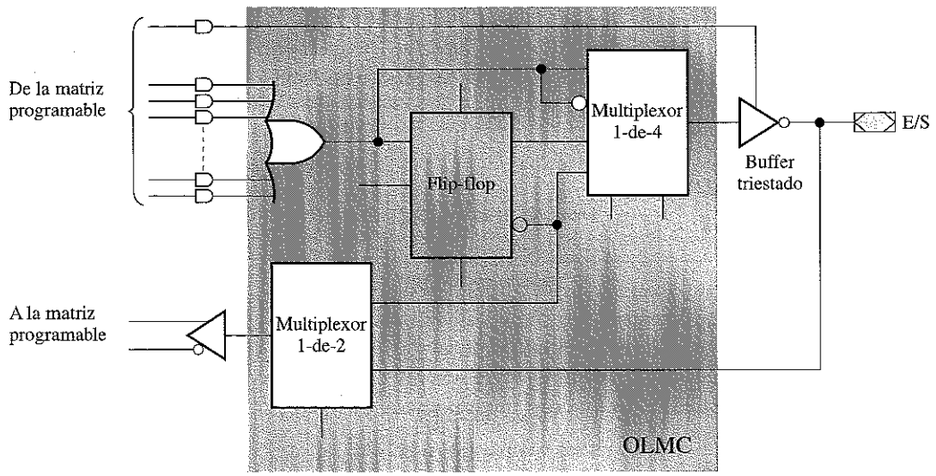


FIGURA 7.20
La OLMC de la GAL22V10.

El multiplexor 1-de-4 conecta una de sus cuatro líneas de entrada al buffer de salida triestado en función del estado de las dos entradas de selección, S_0 y S_1. Las entradas del multiplexor 1-de-4 son la salida de la puerta OR, su complemento, la salida del flip-flop y su complemento. El multiplexor 1-de-2 conecta a través de un buffer a la matriz AND la salida del buffer de tres estados o del flip-flop, en función del estado de la entrada S_1. Los bits de selección S_0 y S_1 de cada OLMC se programan mediante un grupo dedicado de celdas de la matriz.

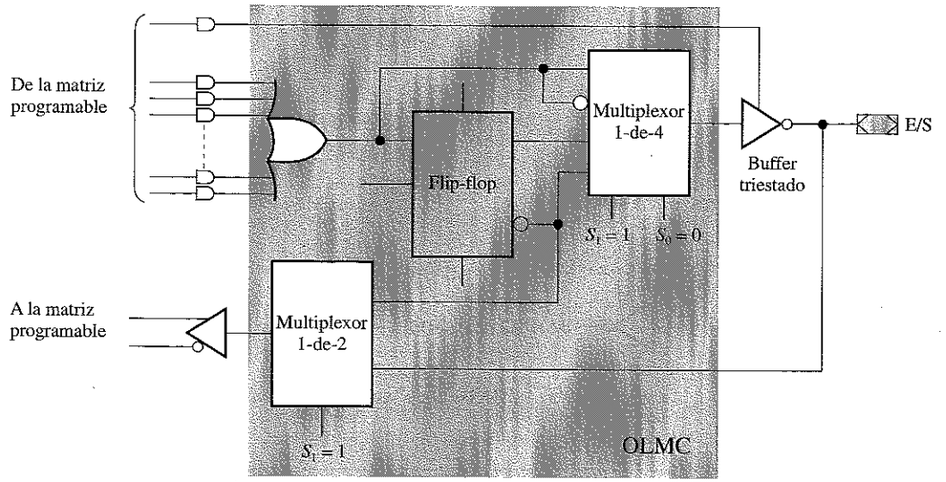
Las cuatro configuraciones de las macroceldas lógicas programables son:

- Modo combinacional con salida activa a nivel BAJO.
- Modo combinacional con salida activa a nivel ALTO.
- Modo secuencial con salida activa a nivel BAJO.
- Modo secuencial con salida activa a nivel ALTO.

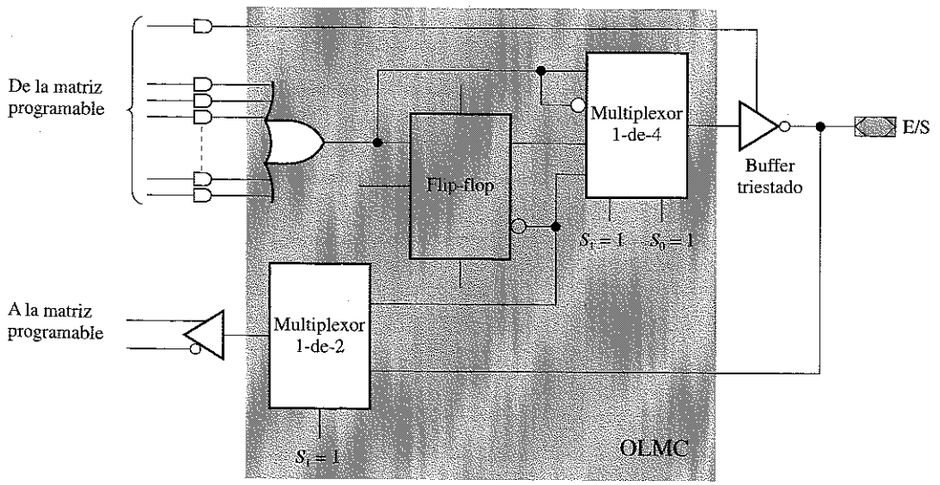
Modo combinacional. Cuando $S_0 = 0$ y $S_1 = 1$, el multiplexor selecciona la salida de la puerta OR. En la Figura 7.21(a) se muestra la lógica *efectiva* de la OLMC, siendo la polaridad de salida activa a nivel BAJO debido a la inversión del buffer de salida triestado. Cuando $S_0 = 1$ y $S_1 = 1$, se selecciona el complemento de la salida de la puerta OR, y la lógica *efectiva* de la OLMC es la que se muestra en la parte (b) de la figura. La salida es activa a nivel ALTO debido a la doble inversión (complemento de OR e inversión del buffer triestado). Los bits S_0 y S_1 de cada una de las diez OLMC se determinan mediante la programación de un conjunto especial de celdas de la matriz, separado de las celdas de la matriz lógica. La OLMC se puede configurar como salida o como entrada controlando el buffer de salida triestado.

Buffer de salida triestado. El **buffer de salida triestado**, mostrado en la Figura 7.22(a) es un inversor de dos estados lógicos, con una línea de control que permite desconectar la salida de la entrada. Los tres estados de salida son: *nivel BAJO* (cuando la entrada está a nivel ALTO), *nivel ALTO* (cuando la entrada está a nivel BAJO) y

alta impedancia (cuando la salida está desconectada de la entrada). Cuando la línea de control está a nivel ALTO, el buffer está activo; y cuando la línea de control está a nivel BAJO, el buffer está en estado de alta impedancia. En la Figura 7.22(b) se ilustran estos tres estados. Obsérvese que, en el estado de alta impedancia, el buffer actúa como un interruptor abierto.



(a) Salida activa a nivel BAJO: $S_1 = 1$, $S_0 = 0$



(b) Salida activa a nivel ALTO: $S_1 = 1$, $S_0 = 1$

FIGURA 7.21
Modo combinacional de la OLMC.

Selección de entrada o de salida. En la lógica en modo combinacional de la OLMC de la Figura 7.21, puede verse que la línea de control del buffer de tres estados resulta de una puerta AND de la matriz. Cuando esta línea de control está a nivel ALTO, la OLMC produce una salida combinacional. Cuando esta línea está a nivel BAJO, el buffer está en abierto (alta impedancia) y el pin funciona como entrada, como se indica en la Figura 7.23. El estado de la línea de control del buffer de tres estados se establece mediante programación.

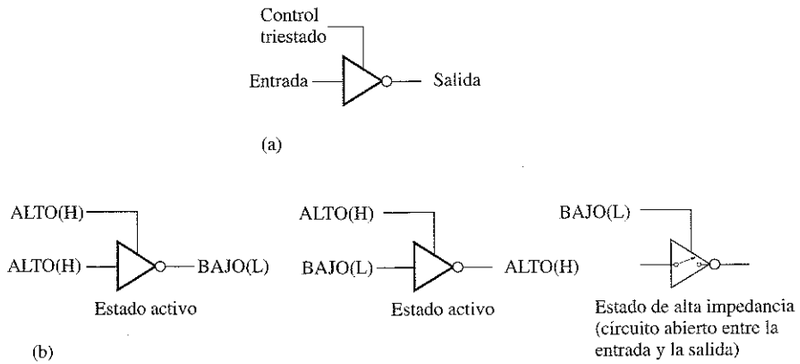
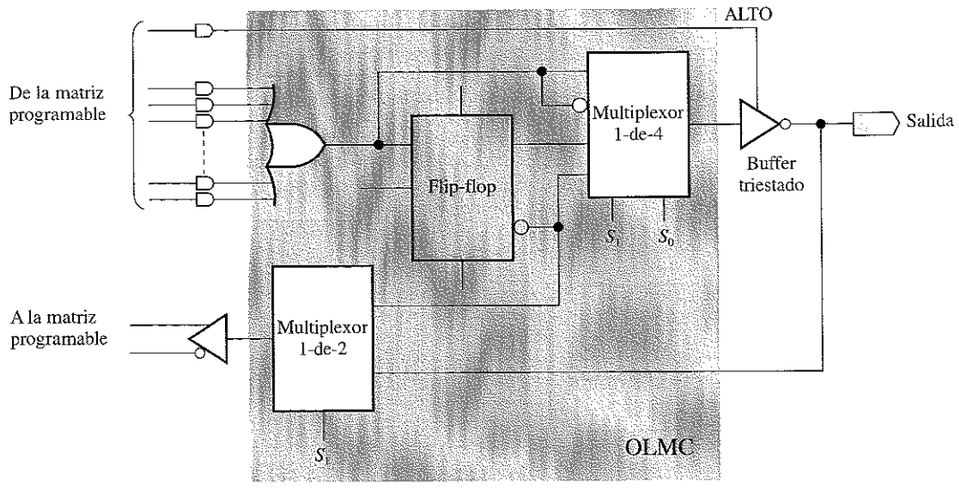
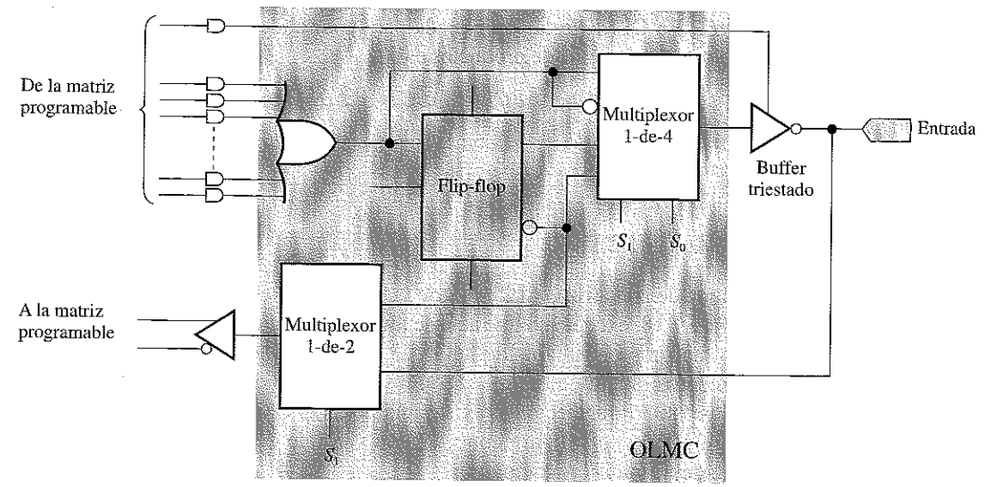


FIGURA 7.22
Funcionamiento del buffer triestado.



(a) Salida (activa a nivel BAJO, en esta configuración)

FIGURA 7.23
Configuraciones de entrada y salida combinacional de la OLMC.



(b) Entrada

FIGURA 7.23
Configuraciones de entrada y salida combinacional de la OLMC. (Continuación).

EJEMPLO 7.5 Determinar la expresión de salida de una OLMC de la GAL22V10, para los productos indicados en la matriz AND y los bits de selección de la Figura 7.24.

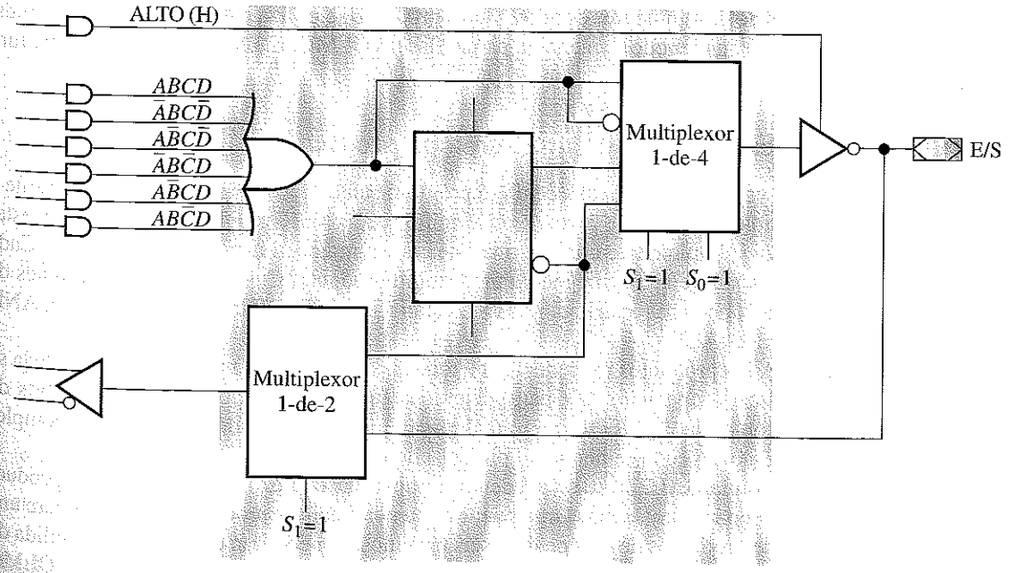


FIGURA 7.24

Solución. La OLMC está configurada para una salida combinacional activa a nivel ALTO; por tanto, la suma de productos es

$$X = ABCD + \overline{A}BC\overline{D} + A\overline{B}C\overline{D} + \overline{A}\overline{B}C\overline{D} + A\overline{B}CD + AB\overline{C}D$$

Problema relacionado. Escribir la suma de productos de salida si $S_0 = 0$ y $S_1 = 1$

La matriz GAL22V10

El diagrama de la matriz GAL22V10 de la Figura 7.25 es similar al diagrama de bloques de la Figura 7.18, excepto en que aquí se muestra en detalle la matriz AND programable.

La matriz programable de la GAL22V10 dispone de 22 líneas de entrada y sus complementos que interseccionan con las 132 líneas de términos producto. Por cada una de las 5808 intersecciones existe una celda E²CMOS. Cada celda se puede programar en estado *activo* para conectar una variable de entrada, o su complemento, a una línea de término producto.

En el diagrama de la Figura 7.25 se representan todos los términos producto mediante una sola línea. Sin embargo, en la GAL22V10 realmente hay 44 líneas para cada puerta AND. Por tanto, cada línea de términos producto está formada por 44 entradas de puerta AND, una para cada línea de entrada y su complemento. Los números mostrados para cada línea de términos producto designan a las 44 celdas asociadas con cada línea. El número de cada OLMC para S_0 y S_1 son los números de las celdas de la matriz en las que se programan estos bits. Estas veinte celdas especiales (de la 5808 hasta la 5827) no se muestran en el diagrama de la matriz.

La Figura 7.26 muestra en detalle la matriz programable correspondiente a la primera OLMC. La OLMC tiene una capacidad de ocho términos producto, que se pueden utilizar en una suma de productos. Uno de los productos se reserva para el control del buffer de tres estados, y otro producto se utiliza para la función reset (puesta a cero) en el modo secuencial, para todas las OLMC. El último producto es una función *global*, ya que es común a todas las OLMC.

Implementación de una suma de productos

Como ya hemos dicho, una celda se programa en estado *activo* cuando una variable o su complemento forma parte de un término producto. El programar una celda en estado *activo* simplemente conecta una línea de entrada con una línea de producto. Cada puerta AND de la matriz genera un término producto. Cuando todas las variables de entrada conectadas a una puerta AND están a nivel ALTO, la salida de la puerta AND es un nivel ALTO.

A los productos procedentes de todas las puertas AND para una OLMC dada se les aplica la operación OR para generar la función suma de productos. Los bits de selección de la OLMC se programan para dirigir la salida de la puerta OR, complementada o sin complementar, al pin de salida.

Con la GAL22V10 se pueden programar hasta diez funciones suma de productos por separado, y la más larga puede contener dieciséis productos. Si se requiere una suma de productos de más de dieciséis términos, se necesitan dos OLMC. La salida de una puerta OR de una OLMC realimenta a la matriz y se conecta a la otra OLMC mediante programación.

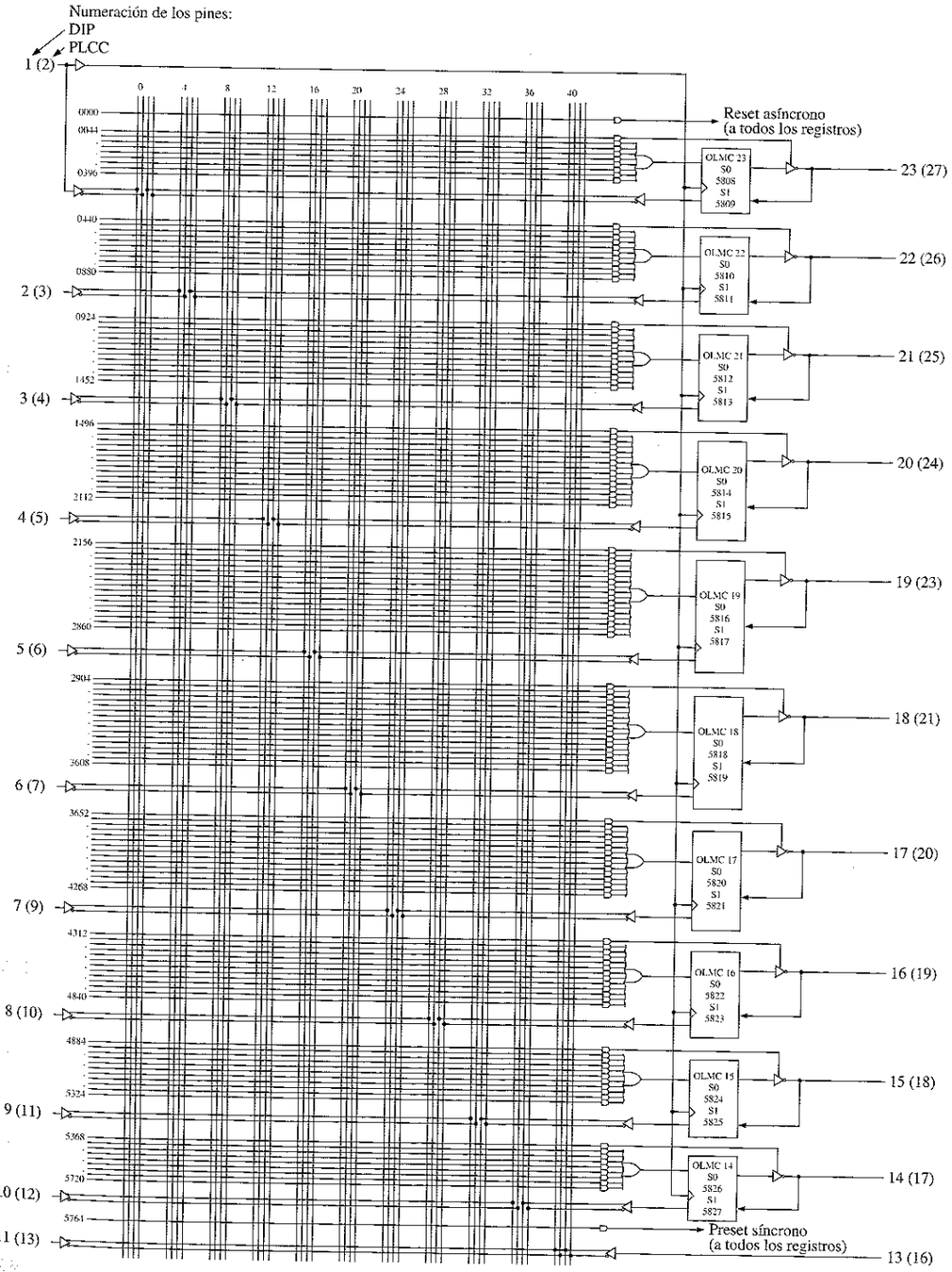


FIGURA 7.25
Diagrama de la matriz GAL22V10.

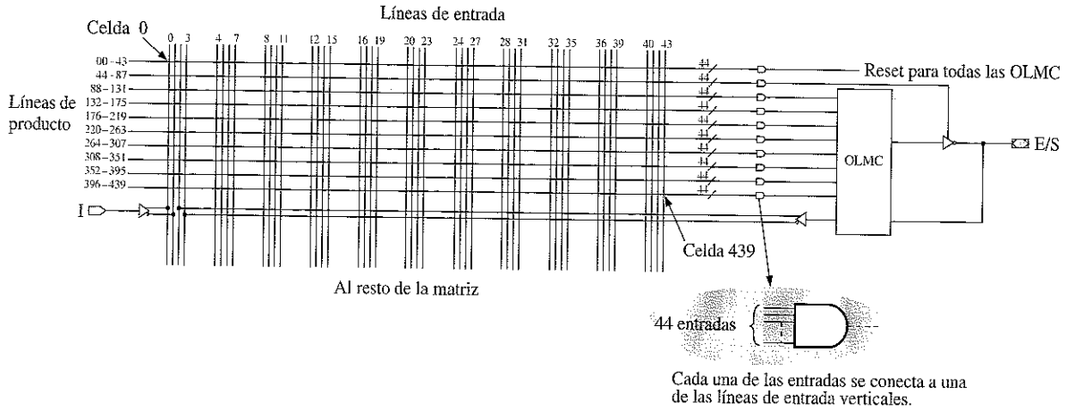


FIGURA 7.26
Organización de la matriz programable para una sección de OLMC.

EJEMPLO 7.6 Demostrar cómo se puede implementar la suma de productos de 6 variables siguiente con la GAL22V10:

$$X = ABCDEF + A\bar{B}\bar{C}D\bar{E}\bar{F} + \bar{A}B\bar{C}D\bar{E}F + \bar{A}BCDE\bar{F} + A\bar{B}\bar{C}\bar{D}\bar{E}F + \bar{A}\bar{B}\bar{C}DEF + \bar{A}\bar{B}\bar{C}\bar{D}\bar{E}\bar{F}$$

Solución. En la Figura 7.27 se muestra la parte de la matriz programada junto con la OLMC asociada. En este caso, el resto de la matriz no se utiliza. Las X representan las celdas E²CMOS que se han programado en estado *activo*.

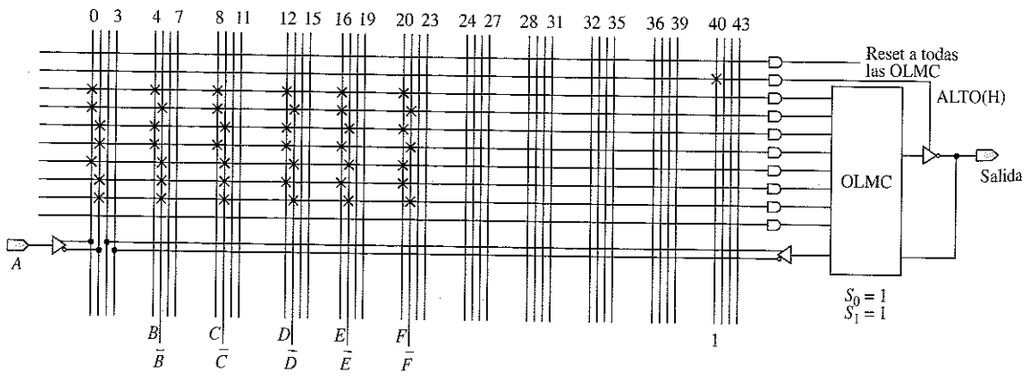


FIGURA 7.27

Problema relacionado. ¿Cuántos términos producto más se pueden añadir a la suma de productos programada de la Figura 7.27?

REVISIÓN DE LA SECCIÓN 7.4

1. ¿Cuáles son los dos modos de configuración de la GAL22V10?
2. ¿Cómo se determina el modo?
3. ¿Cómo se establece la polaridad de la salida?
4. ¿Cuál es el máximo número de términos producto que puede implementarse en una suma de productos con una sola OLMC de una GAL22V10?

7.5 ■ LA GAL16V8

En esta sección se va a tratar otro tipo de matriz lógica genérica, la GAL16V8. La principal característica de la GAL16V8 es que se puede programar para emular muchos tipos de PAL.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Enumerar los modos de la OLMC. ■ Describir el modo simple. ■ Describir el modo complejo. ■ Describir cómo la GAL16V8 puede reemplazar a diversos tipos de PAL.

Diagrama de bloques de la GAL16V8

La GAL16V8 tiene ocho entradas dedicadas y ocho entradas/salidas (E/S), como muestra el diagrama de bloques de la Figura 7.28(a), en el que se indica la numeración de los pines del encapsulado. Este dispositivo está disponible en encapsulado DIP (dual in-line) de 20 pines o en PLCC (plastic chip carrier) también de 20 pines, como muestra la parte (b) de la misma figura. Este dispositivo también está disponible en una versión de baja tensión, la GAL16LV8.

La OLMC

La OLMC de la GAL16V8 es algo diferente de la que incorpora la GAL22V10. La GAL16V8 está diseñada para programarse en uno de los tres modos disponibles para emular a la mayoría de las PAL existentes y, por tanto, puede reemplazar a la PAL para la que se programe.

Emulación de una PAL. Todas las posibles configuraciones de la OLMC de la GAL16V8 se clasifican en tres modos básicos: *simple*, *complejo* y *secuencial*. Cada uno de estos modos permite programar a la GAL16V8 para emular a una serie de dispositivos PAL, que se especifican en la Tabla 7.1.

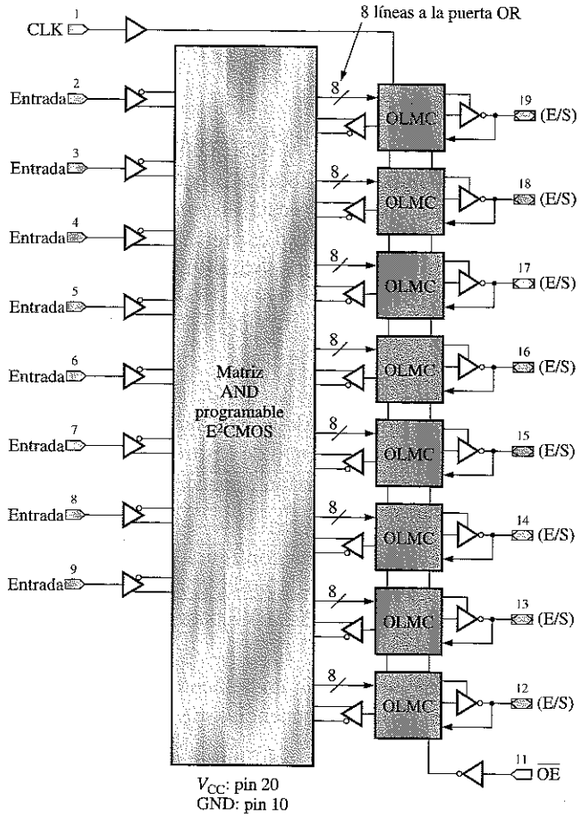
Los modos simple y complejo se asocian únicamente con salidas combinacionales y se explican a continuación. El modo secuencial se verá en el Capítulo 11.

Modo simple. En este modo, las OLMC se configuran como salidas combinacionales dedicadas o como entradas dedicadas (como mucho seis). Las tres posibles configuraciones en este modo son.

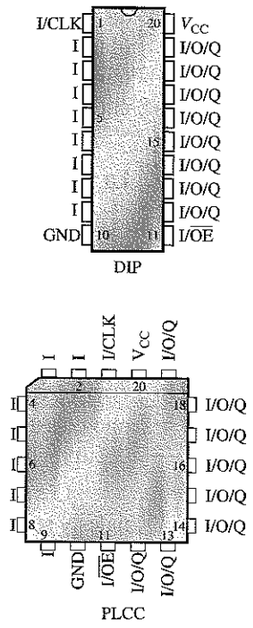
- Salida combinacional.
- Salida combinacional con realimentación a la matriz AND.
- Entrada dedicada.

TABLA 7.1
Dispositivos PAL que se pueden emular para cada modo de las OLMC de la GAL16V8.

GAL16V8 Modo simple	GAL16V8 Modo complejo	GAL16V8 Modo registro
PAL10L8	PAL16L8	PAL16R8
PAL12L6	PAL16H6	PAL16R6
PAL14L4	PAL16P8	PAL16R4
PAL16L2		PAL16RP8
PAL10H8		PAL16RP6
PAL12H6		PAL16RP4
PAL14H4		
PAL16H2		
PAL10P8		
PAL12P6		
PAL14P4		
PAL16P2		



(a)

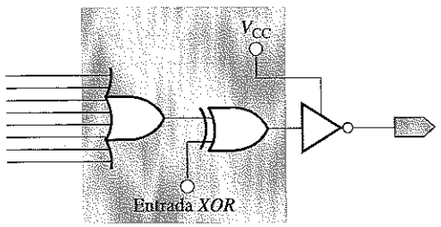


(b)

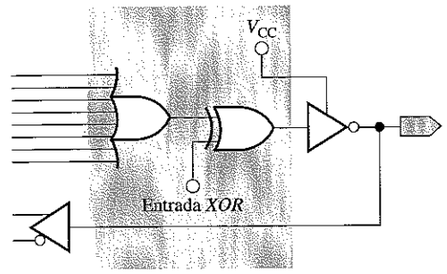
FIGURA 7.28
Diagrama de bloques y encapsulados de la GAL16V8.

Cada una de estas configuraciones se ilustra en la Figura 7.29. Obsérvese que la línea de control triestado está a nivel ALTO (V_{CC}) activando el buffer para las configuraciones de salida combinacional, y a nivel BAJO (a masa) dejando en abierto (alta impedancia) al buffer para la configuración de entrada dedicada.

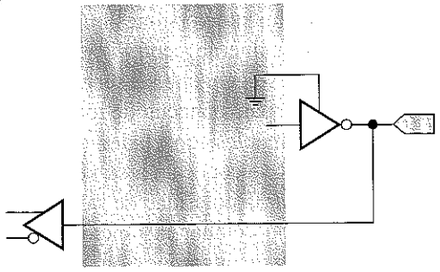
La realimentación permite conectar la salida combinacional a la matriz AND. La entrada *XOR* de la puerta OR-exclusiva determina el estado activo de salida. Cuando *XOR* = 0, la salida es activa a nivel BAJO, ya que la entrada procedente de la puerta OR no resulta invertida. Cuando *XOR* = 1, la salida es activa a nivel ALTO, ya que la entrada procedente de la puerta OR resulta invertida.



(a) Salida combinacional



(b) Salida combinacional con realimentación a la matriz AND



(c) Entrada dedicada

FIGURA 7.29
Configuraciones de la OLMC en el modo simple.

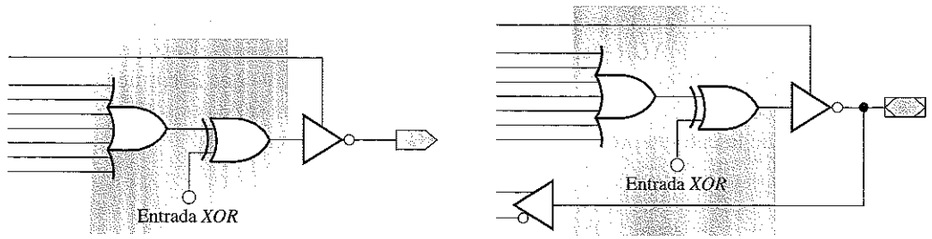
Las ocho entradas de la puerta OR proceden de la matriz AND, e indican que la OLMC puede generar hasta ocho términos producto de una suma de productos.

Modo complejo. En este modo, la OLMC se puede configurar de dos formas:

- Salida combinacional.
- Entrada/salida combinacional.

Estas dos configuraciones se ilustran en la Figura 7.30. En este modo, las entradas/salidas (E/S) están limitadas a seis. Obsérvese que sólo existen siete entradas a la puerta OR procedentes de la matriz AND, ya que la octava entrada se utiliza para el

control del buffer de tres estados. Esto significa que una OLMC puede producir hasta siete términos producto de una suma de productos.



(a) Salida combinacional (b) Entrada/Salida combinacional

FIGURA 7.30
Configuraciones OLMC en el modo complejo.

CPLD y FPGA

Además de los dispositivos PLD tan sencillos como la GAL, hay disponibles otros dispositivos lógicos programables mucho más sofisticados. Los dos tipos básicos son los dispositivos PLD *complejos* (CPLD) y las FPGA (Field Programmable gate array, matrices de puertas programables por campo).

Los CPLD son esencialmente versiones más grandes de los sencillos PLD, con una matriz de interconexión interna centralizada que se utiliza para conectar las macroceldas del dispositivo. Al igual que los sencillos PLD que hemos visto, la matriz de interconexión se programa para conectar de forma selectiva señales de entrada a un nivel de puertas AND programables conectadas a su vez a un nivel de puertas OR fijo. Las salidas de las puertas OR se conectan a su vez a macroceldas configurables, que permiten al usuario especificar la polaridad de salida, seleccionar operaciones combinacionales o de registro, proporcionar la funcionalidad triestado y, opcionalmente, realimentar la señal a la matriz de interconexión. Sin embargo, independientemente de la expresión lógica implementada, el camino a través del CPLD es esencialmente el mismo para cada señal: desde un pin de entrada a través de un buffer o inversor hasta la matriz de interconexión, y de ésta a la lógica AND-OR y de aquí a la macrocelda de salida. Además, las señales de entrada, salida y de reloj se encuentran normalmente en pines específicos del CPLD, por lo que dichos pines son funcionalmente idénticos entre sí. Esta arquitectura limita la flexibilidad del diseño en cierto grado, pero también simplifica el análisis de diseño al ofrecer una temporización de señales desde la entrada hasta la salida muy predecible.

Por otro lado, las FPGA constan de una matriz de gran tamaño de celdas lógicas simples (más simples que las macroceldas de los CPLD) con canales de rutado horizontal y vertical para interconexión. Las FPGA no tienen una estructura de implementación predefinida, como los CPLD, y a través de la cual debe pasar cada señal del CPLD. En lugar de ello, las FPGA permiten que los canales de rutado conecten entre sí las celdas lógicas de múltiples maneras distintas, para implementar las funciones lógicas deseadas. De hecho, y dependiendo de las restricciones de diseño y de los

recursos disponibles, se pueden implementar dos funciones lógicas idénticas de maneras completamente distintas dentro del mismo dispositivo. Este tipo de diseño hace a las FPGA extremadamente flexibles y permite a los diseñadores implementar circuitos lógicos combinatorios y secuenciales muy complejos. Algunas FPGA de gran tamaño pueden incluso ser programadas para operar como microprocesadores.

REVISIÓN DE LA SECCIÓN 7.5

1. Establecer las diferencias básicas entre una GAL16V8 y una GAL22V10.
2. ¿Cómo emula la GAL16V8 un dispositivo PAL?
3. Enumerar los tres modos de la OLMC.

7.6 ■ PROGRAMACIÓN DE LOS DISPOSITIVOS LÓGICOS PROGRAMABLES (PLD)

Como ya hemos visto, las PAL se programan dejando intactos los fusibles especificados y fundiendo los restantes. Las GAL se programan de forma similar, pero son las celdas E²CMOS las que se activan o desactivan. Las funciones lógicas que se implementan determinan las celdas que hay que activar. Para programar una PAL o una GAL se requieren los siguientes elementos: una computadora, un software de programación y un programador de PLD.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Nombrar los componentes necesarios para programar un PLD. ■ Describir el proceso de programación en un organigrama. ■ Definir programación dentro del sistema (In-System Programming, ISP).

Los tres componentes requeridos para programar los PLD son:

1. Software de programación (compilador lógico).
2. Una computadora personal que cumpla los requisitos de software.
3. Un programador controlable por software.

La computadora debe cumplir los requisitos de **software**. El **programador** es un dispositivo hardware que acepta datos de programación procedentes de la computadora e implementa un diseño lógico específico en el PLD, el cual se inserta en un zócalo en el programador. En la Figura 7.31 se ilustran estos elementos.

Computadora

Se puede emplear cualquier computadora personal que cumpla las especificaciones de software y del programador. Normalmente, estas especificaciones incluyen el tipo de microprocesador del que debe disponer la computadora, la cantidad de memoria y el sistema operativo (DOS, Windows y Macintosh son algunos ejemplos).

Software

Los paquetes de software para la programación de dispositivos PLD se denominan **compiladores** lógicos. Existen diversos paquetes de software disponibles, entre los

que se incluyen los siguientes (existen otros además de estos): ABEL, CUPL, OrCAD-PLD y LOG/iC. Todos estos paquetes de software realizan funciones similares: procesan y sintetizan el diseño lógico introducido mediante un método específico, convierten los datos introducidos en un archivo intermedio y luego generan un archivo de salida que se denomina **archivo JEDEC** (conocido también como *mapa de celdas* o *mapa de fusibles*) para el programador del dispositivo. También, con este software, se puede simular y depurar de forma completa un diseño lógico antes de proceder a fabricar el hardware.

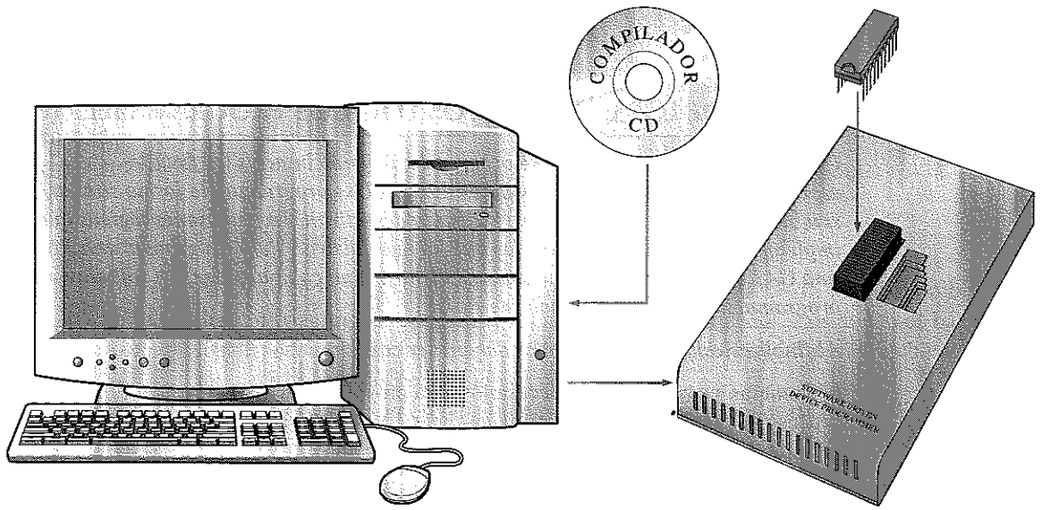


FIGURA 7.31
Componentes para la programación de un PLD.

Una característica fundamental del software para PLD es el método para introducir el diseño lógico. Existen tres métodos básicos de introducir el diseño, que son: *ecuación booleana*, *tabla de verdad* y *máquina de estados*. Existen otros métodos de tipo esquemático que son los diagramas de tiempos y la descripción hardware. Todos los paquetes de software disponibles incluyen varios o todos los posibles métodos de introducción del diseño.

El programador

El dispositivo PLD se inserta en el zócalo del programador, que usualmente es un **zócalo ZIF** (zero insertion force, fuerza de inserción nula). El programador contiene un programa controlador de software que lee el archivo JEDEC generado por el compilador lógico y lo convierte en instrucciones que aplican las tensiones requeridas en pines específicos del PLD, para alterar así las celdas especificadas de la matriz de la forma que indique el mapa de fusibles. Todos los paquetes de software y programadores de PLD, independientemente de los fabricantes, son conformes a un estándar

para la generación del archivo JEDEC establecido por el Joint Electronic Device Engineering Council (JEDEC).

El proceso de programación

El organigrama general de la Figura 7.32 muestra la secuencia para programar un PLD. En primer lugar, se diseña el circuito lógico y se expresa mediante ecuaciones booleanas, tablas de verdad, esquemas lógicos o cualquier otro formato aceptado, y se carga el software en la computadora.

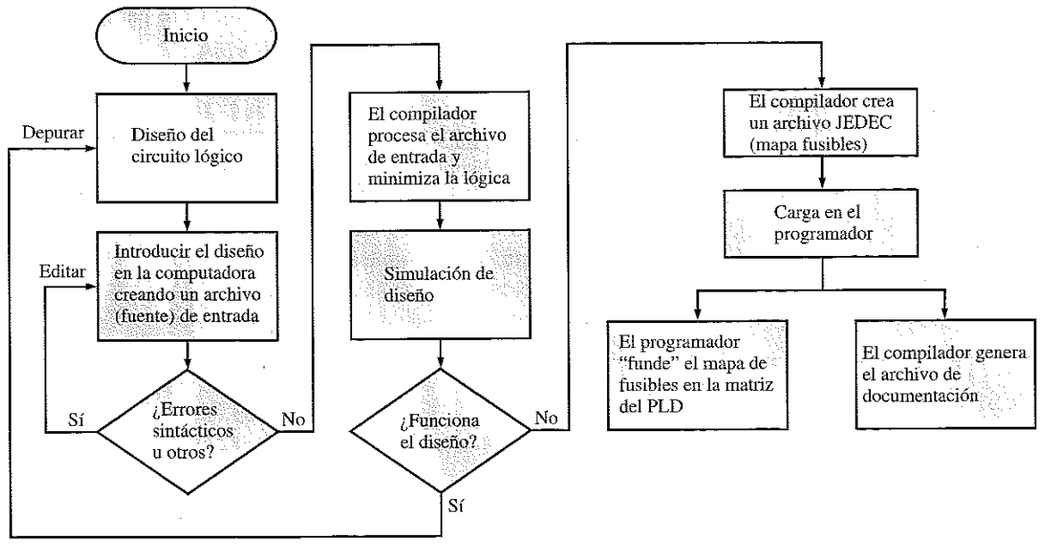


FIGURA 7.32
Organigrama de una secuencia de programación de un PLD.

Introducción del diseño. El diseño lógico se introduce en la computadora a través de un archivo fuente o de entrada. Normalmente, se introduce alguna información preliminar, tal como el nombre del usuario, la empresa, la fecha y la descripción del diseño. Luego, se introduce el tipo de dispositivo PLD, la numeración de los pines de entrada y salida, y las variables. Por último, se introduce la función o funciones lógicas en forma de ecuación booleana, tabla de verdad, esquema o cualquier otro formato. Cualquier error de sintaxis, u otro tipo de error cometido durante la introducción de datos en el archivo de entrada, será indicado y deberá corregirse. La **sintaxis** es el formato prescrito y la simbología que se usa para describir una categoría de funciones.

Ejecución del software. El compilador de software procesa y traduce el archivo de entrada y minimiza la lógica. Entonces, el diseño lógico se simula utilizando un conjunto de entradas hipotéticas conocidas como *vectores de prueba*. Este proceso "ejercita", de forma efectiva, el diseño por software para determinar si trabaja correctamente, antes de programar realmente el PLD. Si se descubre cualquier defecto de

diseño durante la simulación software, el diseño debe depurarse y modificarse para corregir dicho defecto.

Una vez terminado el diseño, el compilador genera un **archivo de documentación**, que incluye las ecuaciones lógicas finales, el archivo JEDEC y, si se desea, un diagrama del patillaje del PLD. El archivo de documentación, esencialmente, facilita una copia en papel del diseño final.

Programación del dispositivo. Cuando se ha terminado el diseño, el compilador crea un mapa de fusibles (archivo JEDEC) que se carga en el programador. Este archivo dice al programador qué fusibles hay que fundir y qué fusibles quedan intactos o, en el caso de una GAL, qué celdas E^2CMOS se *activan* y cuáles se *desactivan*. Luego, el programador utiliza su propio programa controlador de software para modelar el dispositivo PLD de acuerdo con el mapa de fusibles.

Consejos Prácticos

La mayoría de los programadores de dispositivos PLD proporcionan un zócalo de fuerza de inserción cero (ZIF) que aceptará distintos encapsulados de PLD. Es fundamental que el PLD se inserte en el zócalo ZIF con la orientación correcta y en el extremo correcto del zócalo. Si se inserta incorrectamente, el programador puede resultar dañado. Muchos programadores proporcionan un diagrama de inserción junto al zócalo como guía para la apropiada inserción del dispositivo PLD.

En la Sección 7.7 se proporciona una introducción a la programación de los PLD con el software ABEL.

Programación interna del sistema. Los PLD estándar, incluyendo EEPROM, PAL y GAL se programan utilizando un programador hardware, como acabamos de ver. Ciertos dispositivos programables, incluyendo las GAL, también pueden emplearse como dispositivos de programación dentro del sistema (ISP, In-System Programmable); por ejemplo, una versión para programación dentro del sistema de la GAL22V10 se denomina con la referencia ispGAL22V10. El dispositivo ISP es igual que el dispositivo interno estándar, excepto que tiene una lógica adicional para la programación dentro del sistema.

Un dispositivo ISP puede programarse como un dispositivo estándar antes de colocarse en una tarjeta de circuito impreso, o puede programarse de varias formas después de haberse montado en el circuito impreso. Los dispositivos ISP pueden programarse junto con otros circuitos dentro de un sistema de las formas siguientes:

- Utilizando un programador antes de montarlo en la tarjeta de circuito impreso.
- Utilizando un equipo automático de pruebas (ATE, Automatic Test Equipment) después de montarlo en la tarjeta.
- Utilizando una computadora personal después de montarlo en la tarjeta.
- Utilizando un microprocesador interno del sistema (incorporado) después del montaje en la tarjeta.

La programación dentro del sistema (ISP) tiene varias ventajas frente a la programación tradicional de dispositivos PLD. Algunas de estas ventajas son las siguientes:

1. El método ISP permite que los diseños lógicos sean programados y modificados sin tener que retirar el dispositivo de la tarjeta de circuito impreso.
2. Con los PLD tradicionales, es necesaria la manipulación manual tanto en la etapa de diseño y depuración como en la de fabricación. Los frágiles terminales, especialmente de los PLD LSI, que tienen un espesor de 0,5 mm, pueden curvarse fácilmente en el zócalo del programador pudiendo resultar dañados. Los encapsulados para dispositivos ISP se pueden soldar directamente sobre la tarjeta de circuito impreso antes de ser programados y depurados.
3. Se pueden implementar múltiples diseños con un diseño de tarjeta de circuito impreso genérico.
4. Las actualizaciones se pueden realizar en campo, cargando una nueva configuración de circuito desde un disco o descargándolas a través de un módem.

REVISIÓN DE LA SECCIÓN 7.6

1. ¿Qué se requiere para programar un PLD?
2. ¿Cómo se llaman los paquetes de software para dispositivos PLD?
3. ¿Qué es un archivo JEDEC?
4. Definir ISP.
5. ¿En qué difiere un dispositivo ispGAL de una GAL estándar?

7.7 ■ SOFTWARE DE LOS DISPOSITIVOS PLD

Como ya hemos comentado, existen diversos paquetes de software para implementar los diseños lógicos basados en los PLD. ABEL y CUPL son dos de los lenguajes de descripción hardware (HDL, Hardware Description Language) más comúnmente utilizados. Puesto que ambos lenguajes son similares y producen el mismo resultado en términos de programación de un PLD, a menudo, su utilización es una cuestión de preferencias y disponibilidad. Esta sección introduce los conceptos básicos del software para PLD con el lenguaje ABEL y no pretende ser una exposición exhaustiva. Para ampliar información sobre este tema, se recomienda el libro "Digital Design Using ABEL" de Pellerin y Holley (Prentice Hall).

Al finalizar esta sección, el lector deberá ser capaz de:

■ Enumerar los tres formatos para la introducción del diseño. ■ Definir *vector de prueba*. ■ Identificar los símbolos de ABEL para los operadores lógicos. ■ Expresar las funciones lógicas utilizando el formato de ecuación de ABEL. ■ Expresar las funciones lógicas utilizando el formato de tabla de verdad de ABEL. ■ Desarrollar vectores de prueba para un determinado diseño lógico. ■ Escribir un archivo de entrada en ABEL para un determinado diseño lógico. ■ Describir un archivo de documentación. ■ Describir el archivo JEDEC.

Introducción al lenguaje ABEL

ABEL, que es el acrónimo de *Advanced Boolean Expression Language*, permite implementar diseños lógicos en dispositivos lógicos programables. ABEL se puede utilizar para programar cualquier tipo de PLD y, por tanto, es un lenguaje independiente del dispositivo. Como se explicó en la sección anterior, el lenguaje ABEL se ejecuta en una computadora conectada a un programador de dispositivos, independiente del lenguaje, en el que se inserta el PLD.

Introducción del diseño lógico. ABEL proporciona tres diferentes formatos para describir e introducir un diseño lógico desde el teclado de una computadora: *ecuaciones*, *tablas de verdad* y *diagramas de estado*. Las ecuaciones y las tablas de verdad se usan en los diseños lógicos combinacionales, que son en los que se centra este capítulo. El diagrama de estados, así como los otros dos formatos, se pueden utilizar para el diseño de lógica secuencial (máquinas de estados), que no se estudian hasta el Capítulo 11.

Simulación del diseño. Una vez que se ha introducido el diseño lógico, se puede simular su funcionamiento utilizando **vectores de prueba**, para asegurarse de que no existen errores en el diseño. Básicamente, una sección de vectores de prueba enumera todos los posibles valores de entrada y los correspondientes valores de salida. El software, esencialmente, "ejercita" el diseño lógico para asegurarse de que trabaja como se esperaba, tomando todas las posibles combinaciones de entrada y comprobando los resultados de salida.

Síntesis lógica. El proceso software que convierte una descripción del circuito en formato de ecuaciones, tablas de verdad o diagrama de estados, en el archivo JEDEC estándar requerido para implementar realmente el diseño en un PLD se denomina **síntesis** lógica. Usualmente, la síntesis lógica incluye la optimización y minimización del diseño.

Operaciones booleanas

Como muestra la Tabla 7.2, por orden de prioridad, las operaciones NOT (inversión), AND, OR y OR-exclusiva utilizan símbolos especiales en el lenguaje ABEL. Como en el álgebra de Boole, se pueden utilizar paréntesis para alterar el orden de prioridad.

TABLA 7.2
Símbolos ABEL para las operaciones lógicas.

Operación lógica	Símbolo ABEL
NOT	!
AND	&
OR	#
XOR	$

A continuación se detallan ejemplos de las operaciones lógicas en notación estándar booleana y en lenguaje ABEL:

Notación booleana estándar	*ABEL*
$\overline{A}$	!A
AB	A & B
$A + B$	A # B
$A \oplus B$	A $ B

Ecuaciones

Uno de los métodos de entrada de ABEL son las ecuaciones lógicas. El signo igual (=) es el mismo que en las ecuaciones estándar y, cualquier letra o combinación de letras y números, puede emplearse como identificadores de variables. Respecto a los identificadores de las variables, ABEL diferencia entre mayúsculas y minúsculas. Por ejemplo, en ABEL la letra mayúscula *A* y la minúscula *a* son dos variables diferentes.

Todas las ecuaciones en ABEL deben terminarse con un punto y coma (;). Por ejemplo, la suma de productos $X = AB + \overline{A}B + A\,\overline{B}$ se escribe en lenguaje ABEL del siguiente modo:

```
X = A & B # !A & B # A & !B;
```

EJEMPLO 7.7 Escribir cada una de las siguientes expresiones lógicas en lenguaje ABEL:

(a) $X = A\overline{B}C + \overline{A}\,\overline{B}\,\overline{C} + AB + \overline{B}C$ **(b)** $Y = (\overline{A} + B + \overline{C} + D)(A + B + C)$

Solución

(a)
```
X = A & !B & C # !A & !B & !C # A & B # !B & C;
```
(b)
```
Y = (!A # B # !C # D) & (A # B # C);
```

Obsérvese que los paréntesis en la parte (b) se usan para cambiar el orden de prioridad definido en la Tabla 7.2. Es decir, primero se aplica la operación OR a las variables entre paréntesis y luego se aplica la operación AND a los dos términos resultantes.

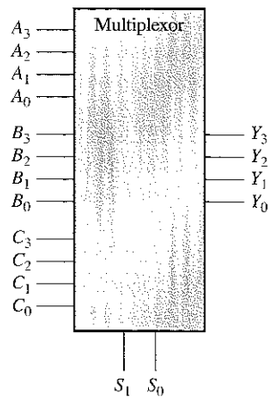
Problema relacionado. Expresar $W = X + Y$ en formato ABEL, siendo $X = \overline{A}\,B\,\overline{C}$ e $Y = A + \overline{B} + C$.

Conjuntos. En algunos casos, las variables de entrada o salida múltiples se pueden agrupar como un conjunto utilizando corchetes, para simplificar una ecuación. Por ejemplo, un grupo de entradas denominadas A_3, A_2, A_1, A_0, pueden definirse como una única variable *A*, utilizando la notación de conjuntos de ABEL del siguiente modo:

```
A = [A3, A2, A1, A0];
```

EJEMPLO 7.8 Desarrollar las ecuaciones lógicas que describen al multiplexor de 12 entradas y 4 salidas de la Figura 7.33(a) utilizando el lenguaje ABEL. Las entradas de selección, S_0

y S_1, seleccionan uno de los tres conjuntos de cuatro entradas de datos, y lo dirigen a las cuatro líneas de salida. La tabla de la función se presenta en la parte (b) de la misma figura.



(a) Diagrama de bloques

Entradas de selección		Salidas de datos			
S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	1	A_3	A_2	A_1	A_0
1	0	B_3	B_2	B_1	B_0
1	1	C_3	C_2	C_1	C_0

(b) Tabla de función

FIGURA 7.33

Solución. Las sumas de productos en formato booleano son:

$$Y_3 = A_3\overline{S}_1S_0 + B_3S_1\overline{S}_0 + C_3S_1S_0$$
$$Y_2 = A_2\overline{S}_1S_0 + B_2S_1\overline{S}_0 + C_2S_1S_0$$
$$Y_1 = A_1\overline{S}_1S_0 + B_1S_1\overline{S}_0 + C_1S_1S_0$$
$$Y_0 = A_0\overline{S}_1S_0 + B_0S_1\overline{S}_0 + C_0S_1S_0$$

Las ecuaciones en lenguaje ABEL son:

```
Y3 = A3 & !S1 & S0 # B3 & S1 & !S0 # C3 & S1 & S0;
Y2 = A2 & !S1 & S0 # B2 & S1 & !S0 # C2 & S1 & S0;
Y1 = A1 & !S1 & S0 # B1 & S1 & !S0 # C1 & S1 & S0;
Y0 = A0 & !S1 & S0 # B0 & S1 & !S0 # C0 & S1 & S0;
```

Las cuatro ecuaciones se pueden combinar en una única ecuación ABEL agrupando las entradas y las salidas en conjuntos con las siguientes declaraciones:

```
A = [A3, A2, A1, A0];
B = [B3, B2, B1, B0];
C = [C3, C2, C1, C0];
Y = [Y3, Y2, Y1, Y0];
S = [S1, S0];
```

Estos conjuntos se usan a continuación para generar una única ecuación de salida del multiplexor:

```
Y = (S == 1) & A # (S == 2) & B # (S == 3) & C;
```

El símbolo == es un operador relacional que indica una comparación de igualdad para los conjuntos. Por ejemplo, el primer término de la derecha de la ecuación es

(S == 1). Esto quiere decir que las entradas de selección S_1 y S_0 son igual a 1, es decir a 01 binario, lo que en notación estándar booleana se expresa como $\overline{S}_1S_0$.

Con las declaraciones de conjuntos y la ecuación única en notación ABEL, el compilador implementará cada una de las cuatro ecuaciones lógicas de origen.

Problema relacionado. Escribir de nuevo la misma ecuación en ABEL si el orden de selección en el multiplexor se invierte. Esto quiere decir que se seleccionan las entradas C cuando las entradas de selección son igual al binario 01, etc.

Tablas de verdad

Otro método de entrada de ABEL utiliza las tablas de verdad. Una función lógica específica puede pasarse a ecuación, como hemos visto, o a tabla de verdad. En cuanto al resultado, ambos métodos son equivalentes. Usualmente, las tablas de verdad son más adecuadas que las ecuaciones para describir ciertos tipos de circuitos, tales como decodificadores.

La tabla de verdad en ABEL se define mediante una cabecera y las entradas de la tabla. El formato de la cabecera se especifica como se indica en la siguiente expresión:

```
TRUTH_TABLE ([A, B, C, D] -> [X1, X2])
```

donde A, B, C y D son las entradas y X1 y X2 son las salidas; el símbolo —> es un operador que indica las entradas que producen las salidas combinacionales. Para ilustrar cómo se elabora una tabla de verdad en lenguaje ABEL, vamos a dar un sencillo ejemplo de una puerta OR-exclusiva con entradas A y B y salida X:

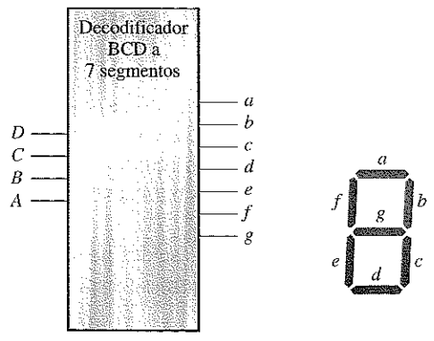
```
TRUTH_TABLE ([A, B] -> [X])
            [1, 0] -> [1];
            [0, 1] -> [1];
            [0, 0] -> [0];
            [1, 1] -> [0];
```

EJEMPLO 7.9 Desarrollar una tabla de verdad ABEL para un decodificador BCD-7 segmentos. Utilizar la tabla de verdad mostrada en la Tabla 4.10. El diagrama de bloques y el formato de 7 segmentos se muestran en la Figura 7.34. Las salidas son activas a nivel ALTO.

Solución. La entrada más significativa es D y la entrada menos significativa es A. Las salidas de los segmentos son de a hasta g. Una salida a 1 significa que el segmento está *encendido* y un 0 significa que el segmento está *apagado*. La tabla de verdad en lenguaje ABEL se escribe:

```
TRUTH_TABLE  ([D, C, B, A] -> [a, b, c, d, e, f, g])
             [0, 0, 0, 0] -> [1, 1, 1, 1, 1, 1, 0];
             [0, 0, 0, 1] -> [0, 1, 1, 0, 0, 0, 0];
```

```
[0, 0, 1, 0] -> [1, 1, 0, 1, 1, 0, 1];
[0, 0, 1, 1] -> [1, 1, 1, 1, 0, 0, 1];
[0, 1, 0, 0] -> [0, 1, 1, 0, 0, 1, 1];
[0, 1, 0, 1] -> [1, 0, 1, 1, 0, 1, 1];
[0, 1, 1, 0] -> [1, 0, 1, 1, 1, 1, 1];
[0, 1, 1, 1] -> [1, 1, 1, 0, 0, 0, 0];
[1, 0, 0, 0] -> [1, 1, 1, 1, 1, 1, 1];
[1, 0, 0, 1] -> [1, 1, 1, 1, 0, 1, 1];
```



(a) Diagrama de bloques (b) Formato del display 7 segmentos

FIGURA 7.34

Utilizando una declaración de conjunto para las entradas y números decimales para los valores de la entrada BCD, la tabla puede simplificarse

```
BCD = [D, C, B, A]; TRUTH_TABLE (BCD -> [a, b, c, d, e, f, g])
                                 0 -> [1, 1, 1, 1, 1, 1, 0];
                                 1 -> [0, 1, 1, 0, 0, 0, 0];
                                 2 -> [1, 1, 0, 1, 1, 0, 1];
                                 3 -> [1, 1, 1, 1, 0, 0, 1];
                                 4 -> [0, 1, 1, 0, 0, 1, 1];
                                 5 -> [1, 0, 1, 1, 0, 1, 1];
                                 6 -> [1, 0, 1, 1, 1, 1, 1];
                                 7 -> [1, 1, 1, 0, 0, 0, 0];
                                 8 -> [1, 1, 1, 1, 1, 1, 1];
                                 9 -> [1, 1, 1, 1, 0, 1, 1];
```

Problema relacionado. Escribir de nuevo la tabla de verdad en ABEL para salidas activas a nivel BAJO (0).

Vectores de prueba

Como ya sabemos, los vectores de prueba se usan en ABEL para describir las señales de entrada y sus correspondientes salidas, con vistas a realizar la simulación lógica.

En otras palabras, los vectores de prueba examinan el diseño lógico antes de programar el hardware, aplicando todas las posibles combinaciones de entrada y comprobando que las salidas son las correctas.

Los vectores de prueba son prácticamente lo mismo que las tablas de verdad y se introducen con el mismo formato. Por ejemplo, los vectores de prueba del decodificador BCD-7 segmentos son:

```
TEST_VECTORS ([D, C, B, A] -> [a, b, c, d, e, f, g])
             [0, 0, 0, 0] -> [1, 1, 1, 1, 1, 1, 0];
             [0, 0, 0, 1] -> [0, 1, 1, 0, 0, 0, 0];
             [0, 0, 1, 0] -> [1, 1, 0, 1, 1, 0, 1];
             [0, 0, 1, 1] -> [1, 1, 1, 1, 0, 0, 1];
             [0, 1, 0, 0] -> [0, 1, 1, 0, 0, 1, 1];
             [0, 1, 0, 1] -> [1, 0, 1, 1, 0, 1, 1];
             [0, 1, 1, 0] -> [1, 0, 1, 1, 1, 1, 1];
             [0, 1, 1, 1] -> [1, 1, 1, 0, 0, 0, 0];
             [1, 0, 0, 0] -> [1, 1, 1, 1, 1, 1, 1];
             [1, 0, 0, 1] -> [1, 1, 1, 1, 0, 1, 1];
```

Los vectores de prueba se pueden simplificar utilizando conjuntos, al igual que las tablas de verdad.

El archivo de entrada ABEL

Cuando se crea un **archivo de entrada** (fuente) en ABEL, se crea un módulo que contiene tres secciones básicas a continuación de las instrucciones para el nombre del módulo y el título: *las declaraciones*, *las descripciones lógicas* y *los vectores de prueba*.

Declaraciones. Generalmente, la sección de declaraciones incluye la *declaración del dispositivo*, las *declaraciones de los pines* y las *declaraciones de conjuntos*. En los Ejemplos 7.8 y 7.9 se han utilizado las declaraciones de conjuntos.

La declaración del dispositivo se utiliza para especificar el PLD que se va a programar, al que algunas veces se denomina *dispositivo objeto* (*target device*). Un ejemplo de una instrucción de declaración de dispositivo es:

```
decoder device 'P22V10';
```

La primera palabra es simplemente una descripción y puede ser cualquier cosa que se desee. La segunda palabra es una palabra clave y puede aparecer en mayúsculas o en minúsculas. El dispositivo debe especificarse de la forma que se indica, utilizando el prefijo P para las PAL y las GAL.

Un ejemplo de instrucción de declaración de pines es:

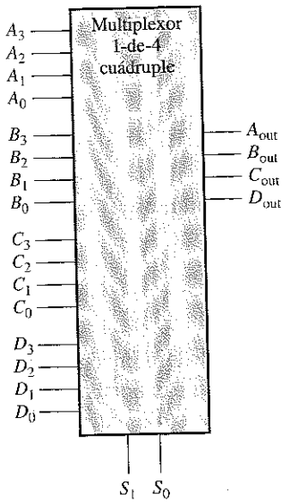
```
A0, A1, A2, A3, PIN 1, 2, 3, 4;
```

La palabra clave PIN debe escribirse en mayúsculas. Los números especificados detrás de esta palabra son los pines del dispositivo PLD que se va a programar. A0 corresponde al pin 1, A1 al pin 2, A2 al pin 3 y A3 al pin 4.

Descripciones lógicas. Los dos métodos de introducción que se han descrito, ecuaciones y tablas de verdad, son dos tipos de descripciones lógicas. Existe un tercer tipo que es el diagrama de estados, que se verá en el Capítulo 11.

Vectores de prueba. Como se ha dicho anteriormente, la sección de los vectores de prueba sirve para comprobar un diseño lógico antes de implementarlo mediante hardware. Los vectores de prueba tienen el mismo formato que las tablas de verdad.

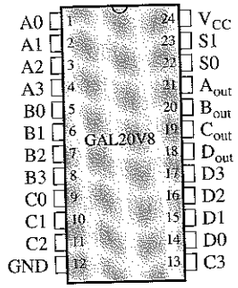
EJEMPLO 7.10 Crear un archivo de entrada en ABEL para implementar el multiplexor cuádruple 1-de-4, que se muestra en el diagrama de bloques de la Figura 7.35(a). En la parte (b) se indica la tabla de verdad, y en la parte (c) el diagrama de asignación de pines del dispositivo objeto GAL20V8.



(a) Diagrama de bloques

Entradas de selección		Salidas de datos			
S_1	S_0	D_{out}	C_{out}	B_{out}	A_{out}
0	0	D_0	C_0	B_0	A_0
0	1	D_1	C_1	B_1	A_1
1	0	D_2	C_2	B_2	A_2
1	1	D_3	C_3	B_3	A_3

(b) Tabla de verdad



(c) Asignación de pines

FIGURA 7.35

Solución. El archivo de entrada en ABEL para el multiplexor es el siguiente:

```
Module quad_1of4_mux

Title              'Quad 1 of 4 multiplexer in a GAL20V8'

        mux      device          'P20V8';
        A0, A1, A2, A3            pin 1, 2, 3, 4;
        B0, B1, B2, B3            pin 5, 6, 7, 8;
        C0, C1, C2, C3            pin 9, 10, 11, 13;
        D0, D1, D2, D3            pin 14, 15, 16, 17;
        Aout, Bout, Cout, Dout    pin 21, 20, 19, 18;
        S0, S1                    pin 22, 23;

Equations
Aout =!S1 & !S0 & A0 # !S1 & S0 & A1 # S1 & !S0 & A2 # S1 & S0 & A3;
Bout =!S1 & !S0 & B0 # !S1 & S0 & B1 # S1 & !S0 & B2 # S1 & S0 & B3;
```



```
Cout =!S1 & !S0 & C0 # !S1 & S0 & C1 # S1 & !S0 & C2 # S1 & S0 & C3;
Dout =!S1 & !S0 & D0 # !S1 & S0 & D1 # S1 & !S0 & D2 # S1 & S0 & D3;

Test_vectors
([S1,S0,A0,A1,A2,A3,B0,B1,B2,B3,C0,C1,C2,C3,D0,D1,D2,D3] ->
[Aout,Bout,Cout,Dout])

"S  S  A  A  A  A  B  B  B  B  C  C  C  C  D  D  D  D        outputs
"1  0  0  1  2  3  0  1  2  3  0  1  2  3  0  1  2  3      A  B  C  D

[0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1] -> [1, 0, 0, 0];
[0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1] -> [0, 1, 0, 0];
[1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1] -> [0, 0, 1, 0];
[1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1] -> [0, 0, 0, 1];

[0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1] -> [1, 1, 1, 0];
[0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1] -> [1, 1, 0, 1];
[1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1] -> [1, 0, 1, 1];
[1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1] -> [0, 1, 1, 1];

END
```

Obsérvese que se pueden insertar líneas de comentarios en cualquier posición, pero deben ir precedidas de dobles comillas (“), como puede verse en la sección de los vectores de prueba, en la que se han incluido para clarificar las entradas y salidas que se introducen. En general, las líneas de comentarios se usan para clarificar o identificar las partes del archivo de entrada, con el propósito de que, posteriormente, cualquiera pueda leer el archivo. El compilador de ABEL ignora las líneas de comentarios y éstas no afectan al diseño final.

Problema relacionado. Utilizar las declaraciones de conjuntos para escribir las ecuaciones lógicas del multiplexor.

El archivo de documentación

Después de que el archivo de entrada se ha procesado, ABEL genera un archivo de documentación. Este archivo proporciona una copia en papel de las ecuaciones finales reducidas, un archivo JEDEC como el que se muestra en la Figura 7.36 y un diagrama de pines del dispositivo.

El archivo JEDEC. Como ya hemos comentado en la Sección 7.6, esta parte del archivo de documentación también se denomina *mapa de celdas* o *mapa de fusibles*. El archivo JEDEC se descarga en el programador para programar el dispositivo objeto. Los archivos JEDEC están estandarizados para todos los lenguajes de programación.

El mapa de celdas muestra aquellas celdas de la matriz del dispositivo que están desconectadas (*desactivadas* o fundidas) y aquellas que están conectadas (*activas* o intactas). Un 0 en un mapa de celdas indica una celda que está conectada y un 1 indica una celda que está desconectada.

```
QP24* QF2706*
L0000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
Lógica de A_out:
11011111011101111111111111111111111111111111
01111110111011111111111111111111111111111111
11110101111011111111111111111111111111111111
11111101010111111111111111111111111111111111
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
Lógica de B_out:
11111110111001111111111111111111111111111111
11111110111011111011111111111111111111111111
11111101111011111111101111111111111111111111
11111101111011111111111011111111111111111111
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
Lógica de C_out:
11111110111011111111111111111111011111111111
11111110111011111111111111111111101111111111
11111101111011111111111111111111111110111111
11111101111011111111111111111111111111101111
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
Lógica de D_out:
11111110111011111111111111111111011111111111
11111110111011111111111111111110111111111111
11111101111011111111111111111101111111111111
11111101111011111111111011111111111111111111
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000*
Configuración de las OLMC y otras programaciones especiales:
L2560
01111000*
L2568
00000000000000000000000000000000000000000000000000000000000000000000*
L2632
1*
L2633
0*
L2634
0*
L2635
0*
L2636
0*
L2637
1*
L2638
1*
L2639
1*
L2640
11111111111111111111111111111111111111111111111111111111111111111111*
L2704
10*
Sección de los vectores de prueba (no programados en la matriz):
V0001 10000100001N00001LLLH00N*
V0002 10000100001N00001LLHL10N*
V0003 10000100001N00001LHLL01N*
V0004 10000100001N00001HLLL11N*
V0005 11101101101N10111LHHH00N*
V0006 11101101101N10111HLHH10N*
V0007 11101101101N10111HHLH01N*
V0008 11101101101N10111HHHL11N*
C5127*
```

FIGURA 7.36
Archivo JEDEC para el multiplexor 1-de-4 cuádruple del Ejemplo 7.10. Las etiquetas grises no forman parte del archivo.

La Figura 7.36 ilustra el mapa de celdas del multiplexor 1-de-4 cuádruple del Ejemplo 7.10. Los cuatro dígitos precedidos por la letra "L" corresponden a los números de línea de producto de la matriz. Las cuatro ecuaciones de la suma de productos de las salidas del multiplexor se implementan en las cuatro secciones de la matriz que se indican. Un asterisco indica el final de la implementación lógica. Todas las líneas que comienzan por "L" son para programación de celdas especiales, para la identificación del usuario, para configurar las OLMC y otras funciones. Las líneas del mapa de celdas que comienzan por "V" se usan para los vectores de prueba, y la última línea que comienza por "C" es una suma de comprobación (checksum).

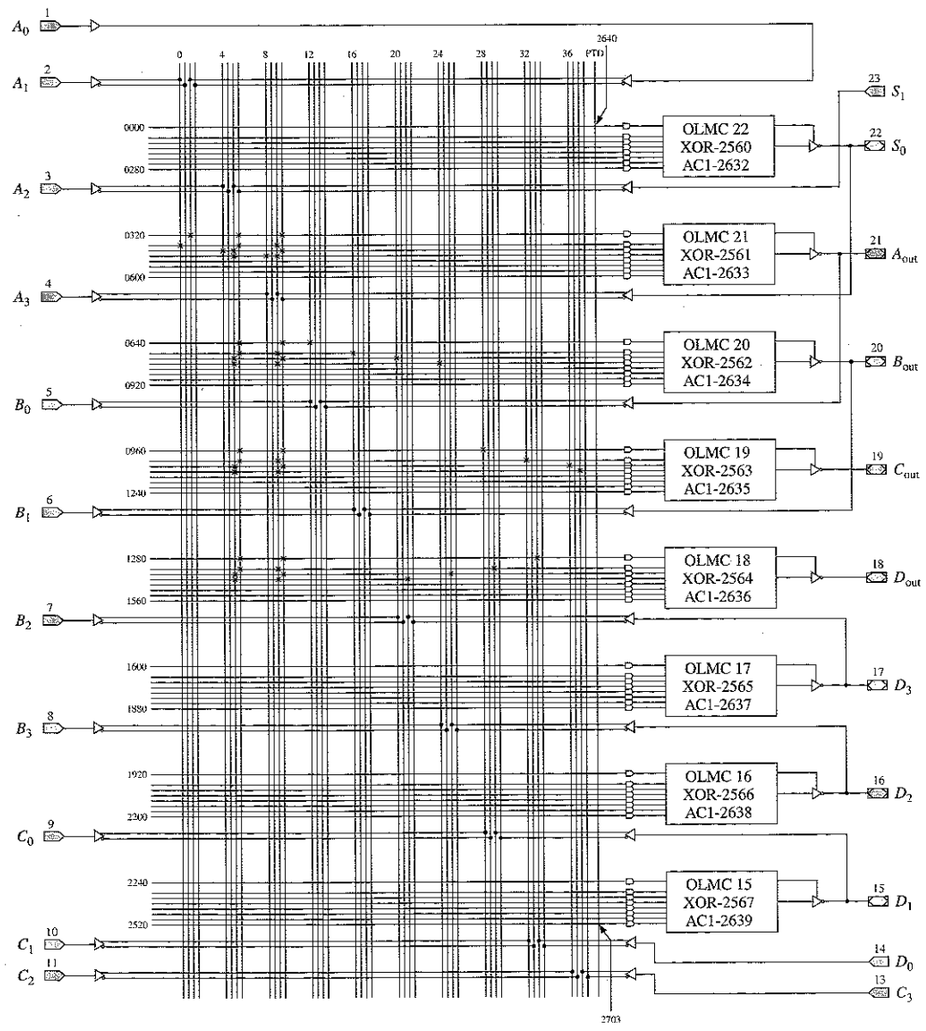


FIGURA 7.37
GAL20V8 para el Ejemplo 7.11.

EJEMPLO 7.11 Utilizando la matriz lógica GAL20V8, hallar la disposición de entradas y salidas y la programación de la matriz para el cuádruple multiplexor 1-de-4, basándose en el archivo JEDEC de la Figura 7.36.

Solución. El diagrama lógico programado para la GAL20V8 se muestra en la Figura 7.37, donde las X indican las celdas *activas*.

Problema relacionado. ¿Cuántos pines de E/S de la GAL20V8 de la Figura 7.37 se utilizan como entradas y cuántos como salidas?

REVISIÓN DE LA SECCIÓN 7.7

1. ¿Qué es ABEL?
2. Enumerar tres formatos para la introducción del diseño en ABEL.
3. Escribir los símbolos en lenguaje ABEL para NOT, AND, OR y XOR.
4. Expresar la siguiente ecuación booleana en lenguaje ABEL:

$$Y = \overline{DEF} + \overline{DE} + F$$

5. ¿Qué son los vectores de prueba?
6. Enumerar las tres partes de un archivo de entrada ABEL.
7. ¿Cuáles son los tipos de declaraciones que usualmente hay en un archivo de entrada?
8. ¿Qué es un archivo de documentación?
9. Describir un archivo JEDEC e indicar otras dos denominaciones que recibe.

7.8 ■ APLICACIÓN A LOS SISTEMAS DIGITALES

En el capítulo anterior, la lógica combinacional del sistema de control de semáforos se implementó utilizando dispositivos SSI y MSI. Ahora, vamos a aplicar nuestros conocimientos sobre dispositivos PLD y la programación de los mismos para implementar la decodificación de estados y la lógica de salida con una GAL22V10. Antes de continuar, debería revisarse la Sección 6.11.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Programar la GAL22V10. ■ Explicar cómo reemplaza esta GAL a la lógica que se implementó en el Capítulo 6.

Decodificación de estados y lógica de salida

En la Figura 7.38 se muestra el diagrama de bloques de la decodificación de estados y la lógica de salida. Como referencia, en la Figura 7.39 se facilita la secuencia de los semáforos y el correspondiente diagrama de estados.

La parte del sistema correspondiente a la decodificación de estados y la lógica de salida tiene dos entradas, S_1 y S_0, y ocho salidas *PR, PA, PV, LR, LA, LV, Largo* y *Corto*. Estas variables de entrada y de salida se definen del siguiente modo:

S_1	Entrada de estado, MSB en código Gray
S_0	Entrada de estado, LSB en código Gray
PR	Salida para la luz roja para la calle principal
PA	Salida para la luz ámbar para la calle principal
PV	Salida para la luz verde para la calle principal
LR	Salida para la luz roja para la calle lateral
LA	Salida para la luz ámbar para la calle lateral
LV	Salida para la luz verde para la calle lateral
Largo	Salida del circuito de disparo para el temporizador largo
Corto	Salida del circuito de disparo para el temporizador corto

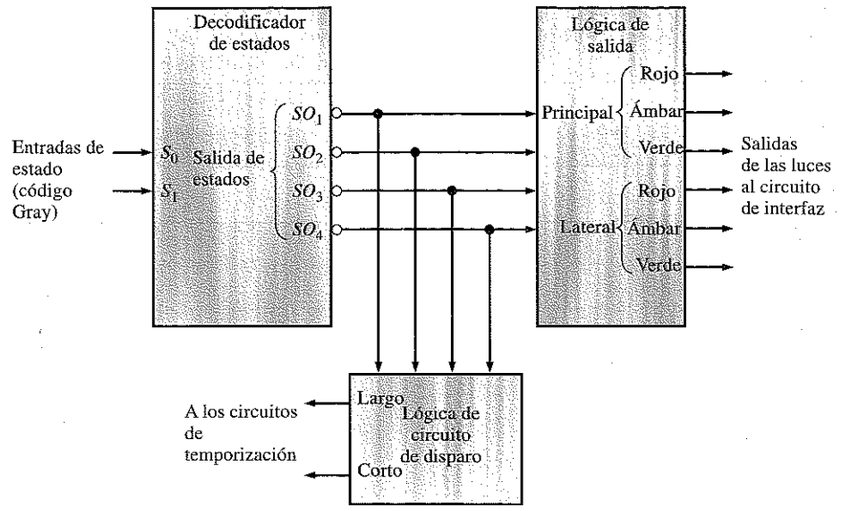
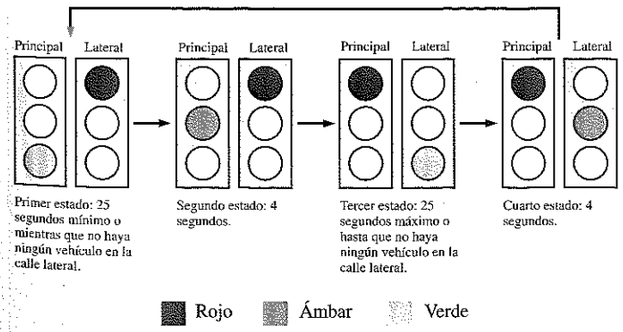
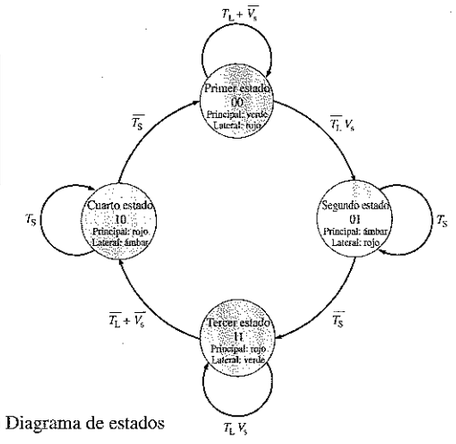


FIGURA 7.38
Diagrama de bloques de la decodificación de estados y de la lógica de salida.



(a) Secuencia de semáforos



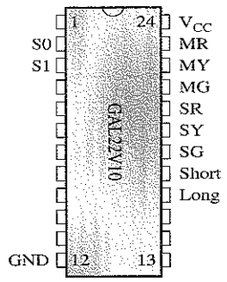
(b) Diagrama de estados

FIGURA 7.39
Secuencia de los semáforos y diagrama de estados.

■ PRÁCTICA DE SISTEMAS 1: Programación de la GAL22V10

- ***Actividad 1***. Utilizando la tabla de verdad y el diagrama de asignación de pines que se muestran a continuación para la GAL22V10, escribir un archivo de entrada en ABEL para programar el dispositivo con la decodificación de estados y la lógica de salida.

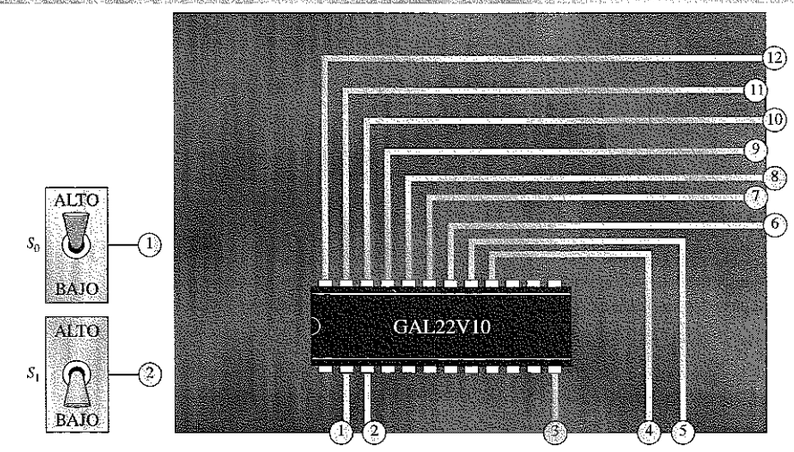
Entradas de estados		Salidas de luces del semáforo						Salidas del circuito de disparo	
S_1	S_0	$\overline{PR}$	$\overline{PA}$	$\overline{PV}$	$\overline{LR}$	$\overline{LA}$	$\overline{LV}$	*Largo*	*Corto*
0	0	1	1	0	0	1	1	1	0
0	1	1	0	1	0	1	1	0	1
1	1	0	1	1	1	1	0	1	0
1	0	0	1	1	1	0	1	0	1



■ PRÁCTICA DE SISTEMAS 2: Verificación y Pruebas

La GAL22V10 en un único chip reemplaza al 74LS139 y al 74LS08, que se utilizaron en el sistema del Capítulo 6 para implementar la misma lógica. Sin embargo, la GAL se puede reprogramar para añadir funciones lógicas requeridas por el sistema, como se verá en el Capítulo 11.

- ***Actividad 1***. Identificar la función de cada uno de los puntos marcados con un círculo y un número en su interior, en la tarjeta de circuito impreso de esta parte del sistema.
- ***Actividad 2***. Utilizando los interruptores conectados como se indica, para simular la secuencia de las entradas en código Gray, desarrollar un procedimiento de prueba para comprobar esta parte de la tarjeta del sistema.



REVISIÓN DE LA SECCIÓN 7.8

1. Después de haber programado la GAL en esta sección, ¿cuántas salidas no se utilizan y quedan disponibles?
2. ¿Cuántas entradas no se utilizan y quedan disponibles?
3. Comparar en términos de complejidad la implementación con la GAL y la implementación hecha en el Capítulo 6.

■ RESUMEN

- Los cuatro tipos básicos de dispositivos PLD se muestran en la Figura 7.40.
- Las PAL son dispositivos programables una única vez.
- Las GAL son dispositivos reprogramables.
- Las PAL y las GAL pueden generar cualquier función lógica suma de productos limitada únicamente por el número de variables.
- Las GAL pueden emular la mayor parte de los tipos de PAL.
- Para programar los dispositivos PLD se utiliza un lenguaje de descripción hardware (HDL), como ABEL o CUPL.

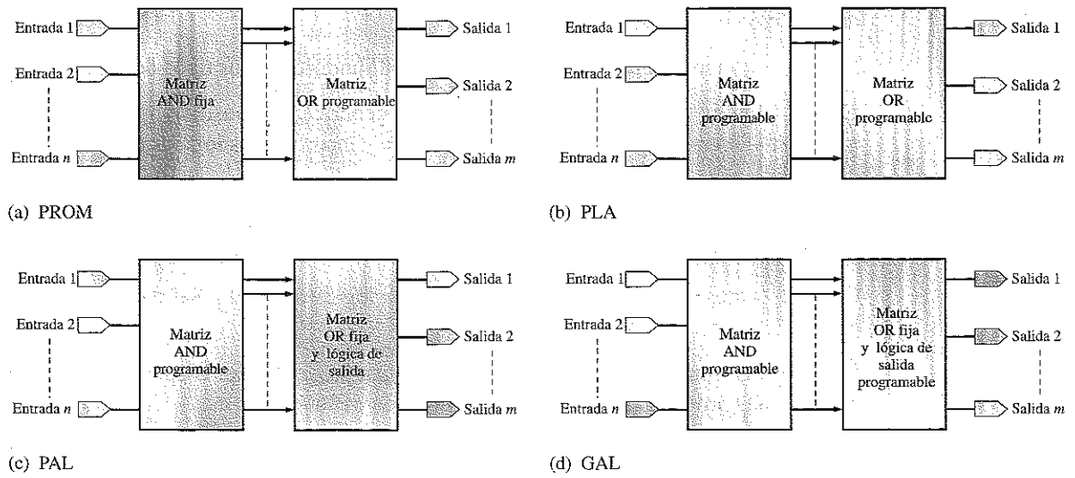


FIGURA 7.40

■ PALABRAS CLAVE

Estos términos también se encuentran en el glosario del final del libro.

ABEL Advanced Boolean Expression Language. Lenguaje de compilación para la programación de PLD; un tipo de lenguaje de descripción hardware (HDL).

Archivo de documentación La información de una computadora que documenta el diseño final posterior al procesamiento del archivo de entrada.

Archivo de entrada La información introducida en una computadora que describe un diseño lógico, utilizando un lenguaje de programación PLD como ABEL o CUPL.

Archivo JEDEC Archivo software estándar generado a partir de un software de compilación, que se emplea en un dispositivo de programación para implementar un diseño lógico en un PLD; también se denomina mapa de fusibles o mapa de celdas.

Arquitectura Disposición funcional interna de los elementos que dotan a un dispositivo de sus características particulares.

Buffer Circuito que evita la carga de una entrada o salida.

Buffer de salida triestado Circuito lógico que posee tres estados de salida: ALTO, BAJO y de alta impedancia (abierto).

Celda Un punto de cruce de una fila y una columna en un PLD.

Compilador Software que traduce el lenguaje de alto nivel que utiliza palabras o símbolos, como ABEL o CUPL, a lenguaje de bajo nivel que entiende la máquina (unos y ceros).

E²CMOS Electrically Erasable CMOS (EECMOS), CMOS eléctricamente borrable. Tecnología de circuitos utilizada para las celdas reprogramables de una GAL.

Entrada/Salida (E/S) Un terminal de un dispositivo que puede ser utilizado como entrada o como salida.

Fusible Elemento que sirve para programar al fundirlo ciertos tipos de PLD.

GAL Generic Array Logic, matriz lógica genérica. Dispositivo que tiene una matriz AND reprogramable, una matriz OR fija y macroceldas lógicas programables de salida.

Matriz En un PLD, una matriz formada por filas de términos productos y columnas de líneas de entrada, con una celda programable en cada unión.

OLMC Output-Logic Macrocell, macrocelda lógica de salida. Salida programable de una GAL.

PAL Programmable Array Logic, matriz lógica programable. Dispositivo con una matriz AND programable y una matriz OR fija.

PLA Programmable Logic Array, matriz lógica programable. Dispositivo con matrices AND y OR programables.

PLD Programmable Logic Device, dispositivo lógico programable.

Programador Instrumento que programa un dispositivo PLD utilizando un archivo JEDEC obtenido de la computadora en el que se esté ejecutando un paquete de software HDL (lenguaje de descripción de hardware).

PROM Programmable read-only semiconductor memory, memoria semiconductora programable de sólo lectura. Dispositivo con una matriz AND fija y una matriz OR programable; se utiliza como dispositivo de memoria, pero, generalmente, no como circuito lógico.

Síntesis Proceso software por el que se convierte una descripción de circuito en un archivo JEDEC estándar para la programación de un PLD.

Software Programa informático. Programas que dicen a la computadora qué es lo que tiene que hacer para poder realizar un determinado número de tareas.

Zócalo ZIF Zócalo de fuerza inserción cero. Tipo de zócalo utilizado en la mayoría de los programadores que aceptan un encapsulado PLD.

■ AUTOTEST

1. Los tipos básicos de matrices programables son
   (a) NOT y XOR **(b)** NAND y NOR
   (c) AND y OR **(d)** diodo y transistor
2. Una matriz puede describirse como
   (a) un grupo de puertas lógicas programables
   (b) una matriz de filas y columnas con una celda programable en cada punto de intersección
   (c) filas de celdas programables
   (d) columnas de fusibles
3. Los dispositivos PLD pueden clasificarse como
   (a) combinacionales y secuenciales
   (b) programables y no programables
   (c) programables una única vez y reprogramables
   (d) PROM, PLA, PAL y GAL
   (e) las respuestas (a) y (c)
   (f) las respuestas (c) y (d)
4. Un PLD que tiene dos matrices programables es
   (a) PAL **(b)** PROM **(c)** PLA **(d)** GAL
5. Un PLD formado por una matriz AND programable y una matriz OR fija es
   (a) PROM **(b)** PLA **(c)** PAL **(d)** GAL
6. Un PLD que tiene una matriz AND reprogramable es
   (a) PROM **(b)** PLA **(c)** PAL **(d)** GAL
7. Los dos tipos de líneas en una matriz PLD son
   (a) vertical y horizontal
   (b) entrada y término producto
   (c) fusible y no fusible
   (d) buffer y de tres estados
8. Una conexión entre una fila y una columna de una matriz PAL se realiza
   (a) fundiendo un fusible
   (b) dejando un fusible intacto
   (c) conectando una variable de entrada a la línea de entrada
   (d) conectando una variable de entrada a la línea de productos
9. La referencia PAL14H4 indica
   (a) una PAL con catorce salidas activas a nivel ALTO y cuatro entradas
   (b) una PAL que implementa catorce puertas AND y cuatro puertas OR
   (c) una PAL con catorce entradas y cuatro salidas activas a nivel ALTO
   (d) quién es el fabricante
10. Una GAL se diferencia de una PAL en que
    (a) una GAL tiene más entradas y salidas
    (b) una GAL se implementa con una tecnología diferente
    (c) una GAL puede reemplazar a diferentes PAL
    (d) una GAL puede reprogramarse y una PAL no

(e) todas las respuestas anteriores
(f) todas las respuestas excepto (a)
(g) todas las respuestas excepto (c)

11. Las celdas reprogramables de una matriz GAL son

(a) TTL (b) E^2CMOS (c) ECL (d) fusibles bipolares

12. OLMC es un acrónimo de

(a) Output Logic Main Cell
(b) Optimun Logic Multiple Channel
(c) Output Logic Macrocell
(d) Odd-parity Logic Master Check

13. Dos formas en las que se puede configurar la salida de una GAL son

(a) combinacional y E/S
(b) simple y complejo
(c) fijo y variable
(d) combinacional y secuencial

14. La referencia GAL22V10 significa

(a) el dispositivo tiene 22 entradas dedicadas y 10 salidas dedicadas
(b) el dispositivo tiene 22 entradas que incluyen entradas dedicadas y E/S y 10 salidas dedicadas o E/S
(c) el dispositivo tiene un número variable de entradas hasta un máximo de 22 y un mínimo de 10

15. Para programar un PLD sólo se necesita

(a) una herramienta especial
(b) una herramienta especial y un PLD maestro que se ha programado en fábrica
(c) una computadora y un programador
(d) una computadora, un programador y un software HDL
(e) una computadora, un programador y software BASIC

16. ISP corresponde a

(a) In-System Programmable
(b) Integrated System Program
(c) Integrated Silicon Programmer

17. ABEL y CUPL son tipos de

(a) programadores
(b) dispositivos PLD
(c) software HDL
(d) disco del sistema operativo

18. Los símbolos para NOT, AND, OR y XOR en ABEL son

(a) !, &, +, $
(b) !, @, #, $
(c) !, &, #, $
(d) *, &, +, !

19. La expresión suma de productos $X = \bar{A}BC + A\bar{B}C + AB\bar{C}$ en ABEL se escribe:

(a) X = !ABC # A!BC # AB!C;
(b) X == !A & B & C # A & !B & C # A & B & !C;
(c) X = !A & B & C # A & !B & C # A & B & !C;
(d) X = (!A & B & C) + (A & !B & C) + (A & B & !C);

20. Un diseño de un circuito lógico que se desea implementar en un PLD se describe, en primer lugar, en

(a) una declaración
(b) un archivo de entrada
(c) un archivo JEDEC
(d) un archivo de documentación

■ PROBLEMAS

SECCIÓN 7.1 Matrices PLD. Clasificación

1. Nombrar y describir cuatro tipos de PLD.

2. Determinar las expresiones booleanas de salida de la matriz AND programada con fusibles de la Figura 7.41.

3. Determinar mediante el número de fila y de columna, los fusibles que deben fundirse en la matriz AND programable de la Figura 7.42, para implementar cada uno de los siguientes términos producto: $X_1 = \bar{A}BC$, $X_2 = AB\bar{C}$, $X_3 = \bar{A}B\bar{C}$.

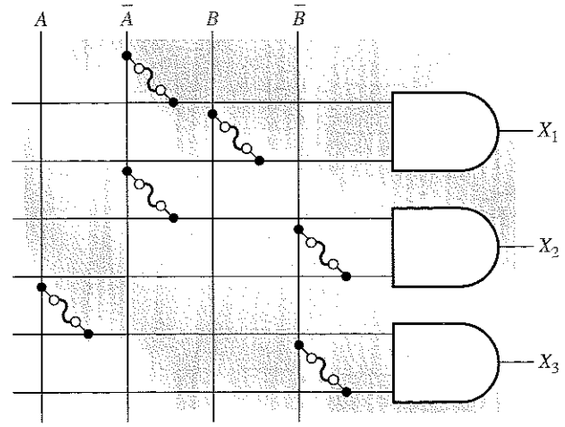


FIGURA 7.41

SECCIÓN 7.2 Matriz lógica programable (PAL, Programmable Array Logic)

4. Determinar la expresión booleana de salida para la PAL mostrada en la Figura 7.43, en la que las X representan los fusibles intactos.

5. Mostrar cómo debería programarse la PAL de la Figura 7.43, para implementar cada una de las siguientes sumas de productos. Utilizar una X para indicar un fusible intacto. Si es necesario, simplificar las expresiones para que se adapten a la PAL indicada.

(a) $Y = A\bar{B}C + \bar{A}B\bar{C} + ABC$
(b) $Y = A\bar{B}C + \bar{A}\bar{B}C + A\bar{B}\bar{C} + \bar{A}BC$

6. Interpretar cada una de las siguientes referencias de dispositivos PAL

(a) PAL16L2 (b) PAL12H6
(c) PAL10P8 (c) PAL16R6

7. Explicar cómo trabaja en una PAL una salida de polaridad programable.

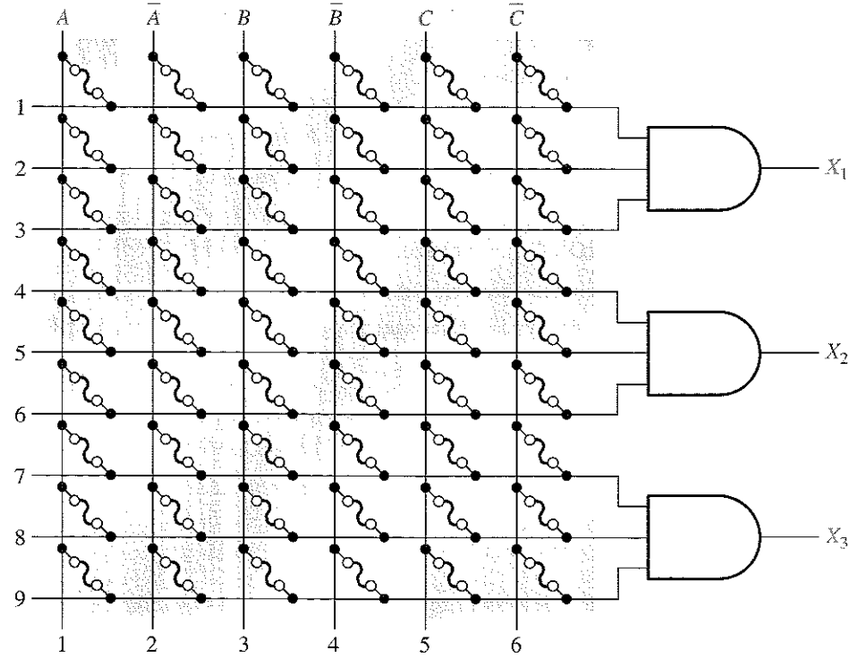


FIGURA 7.42

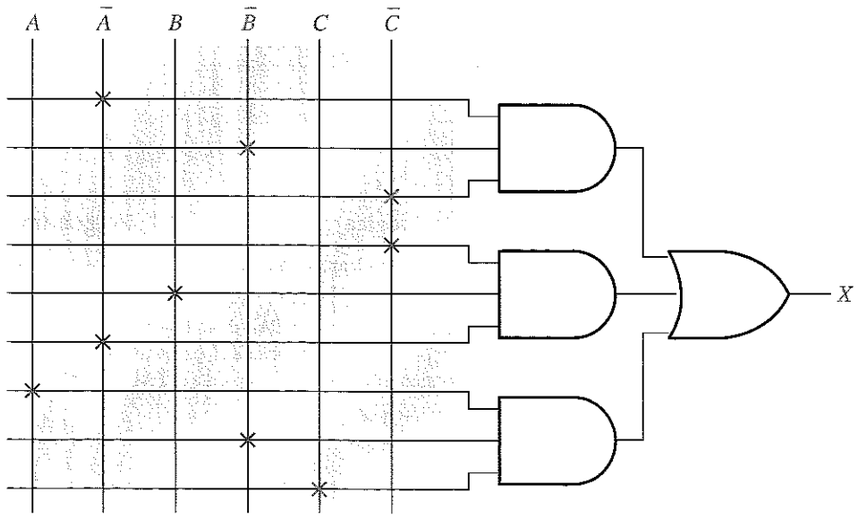


FIGURA 7.43

SECCIÓN 7.3 Matriz lógica genérica (GAL, Generic Array Logic)

8. Determinar la expresión de salida de la matriz GAL mostrada en la Figura 7.44. Las X representan las celdas *activadas*.

9. Indicar cómo debería reprogramarse la GAL de la Figura 7.44 para implementar cada una de las siguientes expresiones. Utilizar X para indicar las celdas *activadas*. Si es necesario, modificar las expresiones:

(a) $X = A\overline{B}C + \overline{A}B\overline{C} + A\overline{B} + BC$ **(b)** $X = (A + \overline{B} + \overline{C})(\overline{A} + B)$

SECCIÓN 7.4 La GAL22V10

10. Describir una GAL22V10 en términos de

(a) número de entradas dedicadas.
(b) número de entradas/salidas (E/S).
(c) número de entradas dedicadas más el número de entradas/salidas (E/S).

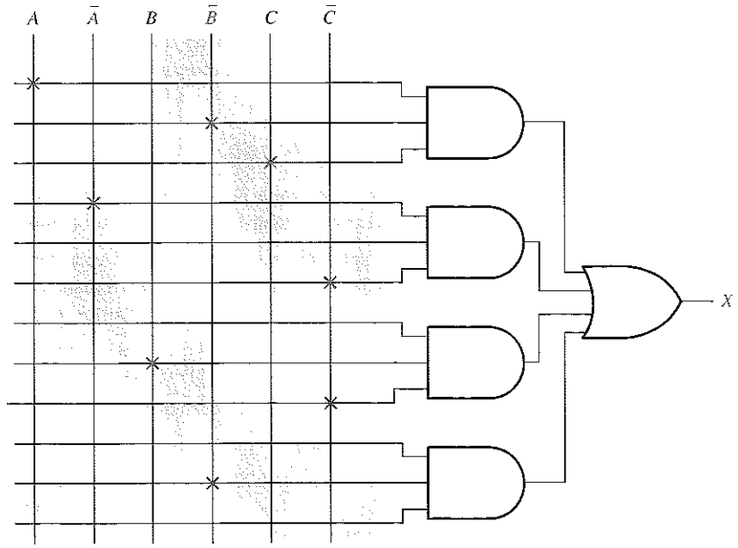


FIGURA 7.44

11. Si se requieren quince entradas en una implementación particular con una GAL22V10, ¿cuál es el número máximo de salidas disponible?

12. ¿Cuál es la mayor expresión suma de productos que se puede implementar con una GAL22V10 en términos de número de variables y de cantidad de términos producto? Suponer que sólo se utiliza una salida.

13. Establecer el propósito de las entradas de selección de cada OLMC en una GAL22V10.

14. Determinar la salida suma de productos de una OLMC en una GAL22V10 para los términos producto y los bits de selección indicados en la Figura 7.45.

SECCIÓN 7.5 La GAL16V8

15. Describir una GAL16V8 en términos de:

(a) número de entradas dedicadas.
(b) número de entradas/salidas (E/S).
(c) número de entradas dedicadas más el número de entradas/salidas (E/S).

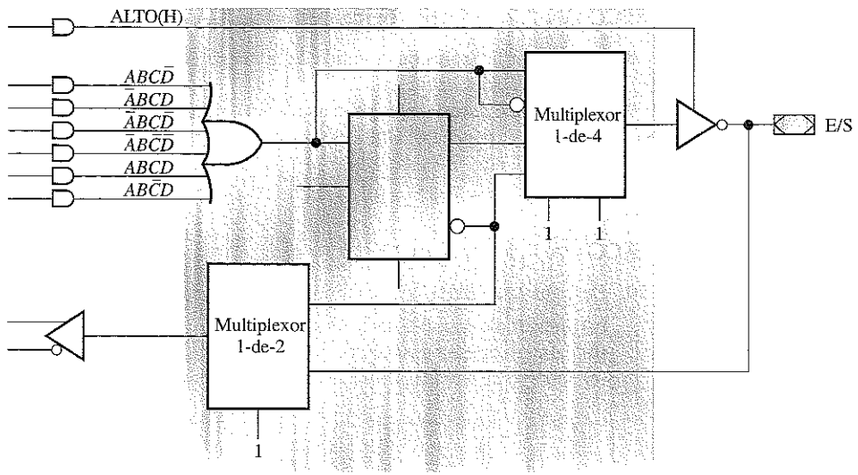


FIGURA 7.45

16. Si hay diez entradas de una GAL16V8, ¿cuál es el número máximo de salidas disponible?

17. ¿Cuál es la mayor expresión suma de productos que se puede implementar con una GAL16V8 en términos de número de variables y de cantidad de términos producto? Suponer que sólo se utiliza una salida.

18. La emulación de una PAL mediante una GAL16V8 se puede conseguir programando el dispositivo en uno de tres posibles modos. ¿Cuáles son estos modos?

SECCIÓN 7.6 Programación de los PLD

19. Enumerar los elementos de los que se debe disponer antes de programar un PLD.

20. Enumerar la secuencia de sucesos que deben ocurrir para programar un PLD.

21. ¿En qué se diferencia una ispGAL22V10 de una GAL22V10?

SECCIÓN 7.7 Software de los PLD

22. Expresar mediante símbolos ABEL cada una de las siguientes operaciones booleanas:

(a) NOT **(b)** AND **(c)** OR **(d)** XOR

23. Convertir a notación ABEL cada una de las siguientes expresiones booleanas:

(a) $\overline{A}\,\overline{B}\,\overline{C}$ **(b)** $A + B + \overline{C}$ **(c)** $A(\overline{B + C})$

24. Determinar la expresión booleana estándar para cada una de las siguientes ecuaciones en ABEL:

(a) W = X & Y & Z # !X & !Y & !Z;

(b) X = !(A & B) # !(!A & B & !C);

(c) !Y = !(A # A & !B # !A & B);

25. Escribir cada una de las siguientes ecuaciones lógicas en ABEL:

(a) $X = A\overline{B}C\overline{D}E + A\overline{B}C + \overline{A}\,\overline{B}$

(b) $Y = AB\overline{C} + DEF + \overline{G}H\overline{I} + J\overline{K}L$

(c) $Z = (\overline{A} + \overline{B} + C + D)(E + \overline{F}) + G$

26. Escribir una declaración de conjunto para las siguientes ecuaciones en lenguaje ABEL, y después expresar el conjunto de ecuaciones en una única ecuación utilizando la notación precisa.

```
X1 = A1 & P1 & Q1 # B1 & !P1 & Q1 # C1 & P1 & !Q1;
X2 = A2 & P1 & Q1 # B2 & !P1 & Q1 # C2 & P1 & !Q1;
X3 = A3 & P1 & Q1 # B3 & !P1 & Q1 # C3 & P1 & !Q1;
```

27. Escribir en ABEL la tabla de verdad de una puerta NOR-exclusiva.

28. Escribir en ABEL la tabla de verdad de un decodificador BCD-decimal con las salidas activas a nivel ALTO.

29. Escribir los vectores de prueba para el decodificador BCD-decimal del problema 28.

Aplicación a sistemas

SECCIÓN 7.8 Aplicación a los sistemas digitales

30. En el sistema de control de semáforos, el intervalo de tiempo más largo aumenta de 25 s a 45 s, y el intervalo de tiempo más corto pasa de 4 s a 5 s. ¿Cómo afecta este cambio a la tabla de verdad de la decodificación de estados y de la lógica de salida?

31. ¿Puede utilizarse una GAL16V8 para implementar la decodificación de estados y la lógica de salida? Razone su respuesta.

Problemas especiales de diseño

32. Escribir en ABEL un archivo de entrada completo para el decodificador BCD-decimal con las salidas activas a nivel ALTO, para implementarse con una GAL22V10.

33. Escribir en ABEL un archivo de entrada para implementar un codificador de prioridad BCD-decimal con salidas activas a nivel ALTO, utilizando una GAL16V8. Suponer que sólo se activará una entrada cada vez.

34. Utilizar una GAL22V10 para reemplazar los tres 74185 utilizados en el convertidor binario-BCD del Capítulo 6 (Figura 6.41). Desarrollar el archivo de entrada. Comentar los posibles problemas de su implementación.

■ RESPUESTAS A LAS REVISIONES DE CADA SECCIÓN

SECCIÓN 7.1

1. Los cuatro tipos de PLD son: PROM, PLA, PAL y GAL.
2. En una PLA, ambas matrices son programables. En una PAL, sólo la matriz AND es programable.
3. Una PAL se programa una única vez. Una GAL es reprogramable y tiene salidas programables.

SECCIÓN 7.2

1. Una PAL es un PLD con una matriz AND programable y una matriz OR fija.
2. El fusible es el elemento programable de una matriz.
3. Cualquier combinación lógica se puede implementar con una PAL, teniendo en cuenta las limitaciones de disponibilidad de las entradas y salidas.
4. Existen tres tipos de lógica de salida de una PAL: salida combinacional, E/S combinacional y salida de polaridad programable.
5. La referencia PAL12H6 indica un dispositivo PAL de 12 entradas, que incluyen las E/S, y 6 salidas activas a nivel ALTO.

SECCIÓN 7.3

1. Una GAL es un PLD con una matriz AND reprogramable, una matriz OR fija y una lógica de salida programable.
2. Una celda E^2CMOS puede programarse repetidas veces, para conectar o desconectar una línea de entrada y una línea de producto.
3. Cualquier lógica combinacional se puede implementar con una GAL, teniendo siempre en cuenta las limitaciones de entradas y salidas disponibles.
4. OLMC (Output Logic Macrocell) es la macrocelda lógica de salida en una GAL. Puede programarse para varias configuraciones de salida.

SECCIÓN 7.4

1. La GAL22V10 dispone de modo combinacional y de modo secuencial.
2. El modo se determina mediante los bits de selección programables.
3. La polaridad se establece mediante los bits de selección programables.
4. Con una OLMC de una GAL22V10 se puede implementar un máximo de dieciséis términos producto de una suma de productos.

SECCIÓN 7.5

1. Una GAL22V10 tiene más entradas y salidas que una GAL16V8, pero la GAL16V8 puede programarse fácilmente para emular muchas PAL.
2. La emulación de una PAL se hace mediante la programación de las OLMC.
3. Los modos de la GAL16V8 son: simple, complejo y secuencial.

SECCIÓN 7.6

1. Para programar un PLD se requiere una computadora, un programador y un software para PLD.
2. El software para PLD son compiladores.
3. Un archivo JEDEC (Joint Electronic Device Engineering Council) es la salida final del software PLD que programa realmente al dispositivo.
4. ISP corresponde a in-system programmable.
5. Una ispGAL puede programarse después de haberse montado en la tarjeta de circuito impreso. Una GAL estándar debe programarse en un programador.

SECCIÓN 7.7

1. ABEL (Advanced Boolean Expression Language) es un lenguaje de descripción hardware para PLD.
2. Los formatos para introducir un diseño son: ecuaciones, tablas de verdad y diagramas de estados.
3. NOT: !, AND: &, OR: #, XOR: $
4. Y = !D & E & !F # !D & !E # F
5. Los vectores de prueba enumeran los valores de entrada y salida para probar un diseño lógico.
6. Las tres partes de un archivo de entrada ABEL son: declaraciones, descripciones lógicas y vectores de prueba.
7. Usualmente, un archivo de entrada contiene una declaración del dispositivo, declaraciones de los pines y declaraciones de conjuntos.
8. Un archivo de documentación es el archivo de salida que contiene las ecuaciones reducidas, el archivo JEDEC y un diagrama de los pines.

9. Un archivo JEDEC es el archivo que se introduce en el programador para programar realmente al PLD. También se denomina mapa de fusibles o mapa de celdas.

SECCIÓN 7.8

1. Hay dos salidas que no se utilizan después de programar la GAL22V10.
2. Hay diez entradas no utilizadas y disponibles.
3. La GAL reemplaza a dos circuitos integrados.

■ RESPUESTAS A LOS PROBLEMAS RELACIONADOS DE LOS EJEMPLOS

7.1 $X = \overline{B}C + \overline{A}B\overline{C} + \overline{A}\,\overline{B} + C$

7.2 Un PAL14H4 tiene 14 entradas y 4 salidas activas a nivel ALTO

7.3 $X = B\overline{C} + BC + BC + A\overline{B} = \overline{B}C + BC + A\overline{B}$

7.4 Una GAL18V10 tiene hasta 18 entradas y 10 salidas.

7.5 $X = \overline{ABCD + \overline{A}BC\overline{D} + A\overline{B}C\overline{D} + \overline{A}B\overline{C}D + A\overline{B}CD + AB\overline{C}D}$

7.6 Se puede añadir un término producto más.

7.7
```
W = !A & B & !C # A # !B # C;
```

7.8
```
Y = (S == 3) & A # (S == 2) & B # (S == 1) & C;
```

7.9
```
TRUTH_TABLE (BCD -> [a, b, c, d, e, f, g])
                0 -> [0, 0, 0, 0, 0, 0, 1];
                1 -> [1, 0, 0, 1, 1, 1, 1];
                2 -> [0, 0, 1, 0, 0, 1, 0];
                3 -> [0, 0, 0, 0, 1, 1, 0];
                4 -> [1, 0, 0, 1, 1, 0, 0];
                5 -> [0, 1, 0, 0, 1, 0, 0];
                6 -> [0, 1, 0, 0, 0, 0, 0];
                7 -> [0, 0, 0, 1, 1, 1, 1];
                8 -> [0, 0, 0, 0, 0, 0, 0];
                9 -> [0, 0, 0, 0, 1, 0, 0];
```

7.10
```
BIT0 = [D0, C0, B0, A0];
BIT1 = [D1, C1, B1, A1];
BIT2 = [D2, C2, B2, A2];
BIT3 = [D3, C3, B3, A3];
   S = [S1, S0];
 OUT = [Dout, Cout, Bout, Aout];
 OUT = (S == 0) & BIT0 # (S == 1) & BIT1 # (S == 2)
                                & BIT2 # (S == 3) & BIT3;
```

7.11 Se usan 18 pines como entradas. Cuatro pines se usan como salidas.

■ RESPUESTAS AL AUTOTEST

1. (c)	**2.** (b)	**3.** (f)	**4.** (c)	**5.** (c)	**6.** (d)	**7.** (b)	**8.** (b)
9. (c)	**10.** (f)	**11.** (b)	**12.** (c)	**13.** (d)	**14.** (b)	**15.** (d)	**16.** (a)
17. (c)	**18.** (c)	**19.** (c)	**20.** (b)				

8

FLIP-FLOPS Y DISPOSITIVOS RELACIONADOS

8.1 Latches
8.2 Flip-flops disparados por flanco
8.3 Flip-flops maestro-esclavo
8.4 Características de operación de los flip-flops
8.5 Aplicaciones de los flip-flops
8.6 Monoestables
8.7 El temporizador 555
8.8 Localización de averías
8.9 Aplicación a los sistemas digitales

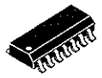
Simulación por computadora de circuitos digitales:
El tutorial de Electronics Workbench (EWB) se encuentra en la dirección http://www.prenhall.com/floyd.

■ OBJETIVOS DEL CAPÍTULO

- ❑ Utilizar puertas lógicas para construir latches básicos.
- ❑ Explicar la diferencia entre un latch S-R y un latch D.
- ❑ Conocer las diferencias entre un latch y un flip-flop.
- ❑ Explicar en qué se diferencian los flip-flops S-R, D y J-K.
- ❑ Explicar en qué se diferencian los flip-flops disparados por flanco y los maestro-esclavo.
- ❑ Comprender el significado de: retardo de propagación, tiempo de establecimiento (*setup time*), tiempo de mantenimiento (*hold time*), frecuencia máxima de funcionamiento, ancho mínimo del impulso de reloj y disipación de potencia en las aplicaciones de los flip-flops.
- ❑ Emplear flip-flops en aplicaciones sencillas.
- ❑ Analizar los circuitos en busca de condiciones de concurrencia (*race conditions*) y la aparición de glitches.
- ❑ Explicar en qué se diferencian los monoestables redisparables y no redisparables.
- ❑ Conectar un temporizador 555 para operar como multivibrador aestable o como monoestable.
- ❑ Métodos de depuración de nuevos diseños.
- ❑ Localizar las averías en circuitos básicos de flip-flops y monoestables.
- ❑ Utilizar monoestables en una aplicación de sistemas.

■ INTRODUCCIÓN

En este capítulo, se van a tratar los circuitos biestables, monoestables y los dispositivos lógicos aestables, denominados *multivibradores*. Los dispositivos biestables se dividen en dos categorías: flip-flops y latches. Los biestables poseen dos estados estables, denominados SET (activación) y RESET (desactivación), en los cuales se pueden mantener indefinidamente, lo que les hace muy útiles como dispositivos de almacenamiento. La diferencia básica entre latches y flip-flops es la manera en que cambian de un estado a otro. Los flip-flops son los bloques básicos de construcción de los contadores, registros y otros circuitos de control secuencial, y se emplean también en ciertos tipos de memorias. El multivibrador monoestable, normalmente denominado monoestable, tiene un único estado estable. Un monoestable genera un único impulso de anchura controlada cuando se activa o dispara. El multivibrador aestable no tiene ningún estado estable y se emplea principalmente como oscilador, es decir, como generador de señales automantenido. Los osciladores de impulsos se emplean como fuentes de señales de temporización en los sistemas digitales.

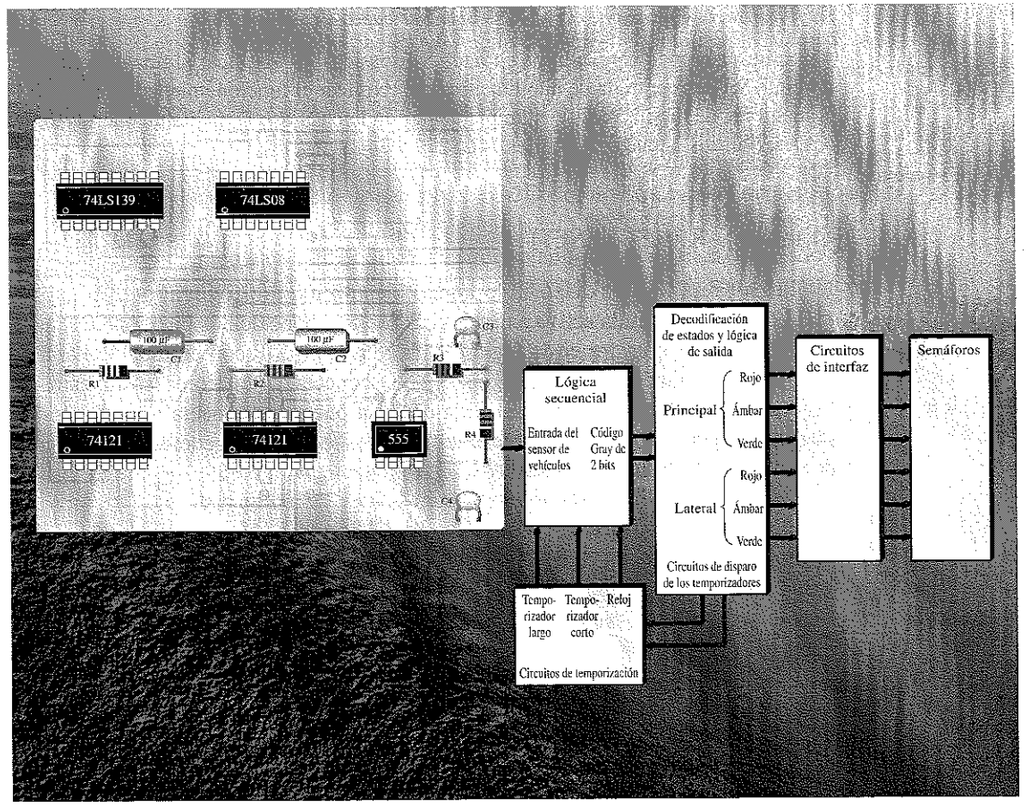


DISPOSITIVOS ESPECÍFICOS

74AHC74	74LS279	74LS122
555	74121	74LS75
74HC112		

■ APLICACIÓN A LOS SISTEMAS DIGITALES

Esta aplicación a un sistema digital ilustra los conceptos que se explican a lo largo del capítulo. La aplicación de la Sección 8.9 continúa con el sistema de control de semáforos del Capítulo 6. Este capítulo se ocupa del circuito de temporización del sistema que genera la señal de reloj, del intervalo de temporización largo para las luces rojas y verdes y del intervalo de temporización corto para las luces ámbar. El reloj se emplea como señal básica de temporización, que hace que la lógica secuencial del sistema pase a través de sus estados. La lógica secuencial se desarrollará en el Capítulo 9.



8.1 ■ LATCHES

El latch (cerrojo) es un tipo de dispositivo de almacenamiento temporal de dos estados (biestable), que se suele agrupar en una categoría diferente a la de los flip-flops. Básicamente, los latches son similares a los flip-flops, ya que son también dispositivos de dos estados que pueden permanecer en cualquiera de sus dos estados gracias a su capacidad de realimentación, lo que consiste en conectar (realimentar) cada una de las salidas a la entrada opuesta. La diferencia principal entre ambos tipos de dispositivos está en el método empleado para cambiar de estado.

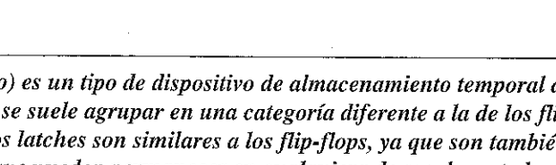
Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar el funcionamiento de un latch S-R básico. ■ Explicar el funcionamiento de un latch S-R con entrada de habilitación. ■ Explicar el funcionamiento de un latch D con entrada de habilitación. ■ Implementar un latch S-R o D mediante puertas lógicas. ■ Describir los latches cuádruples 74LS279 y 74LS75.

NOTAS INFORMÁTICAS
Los dispositivos *latch* algunas veces se utilizan en sistemas informáticos para multiplexar datos sobre un bus. Por ejemplo, los datos que se introducen en una computadora desde una fuente externa tienen que compartir el bus de datos con datos procedentes de otras fuentes. Cuando el bus de datos no está disponible para la fuente externa, los datos existentes deben almacenarse y para ello pueden colocarse *latches* entre la fuente externa y el bus de datos. Cuando el bus de datos no está disponible para la fuente externa, los *latches* deben desconectarse del bus utilizando un método conocido como tri-estado. Cuando el bus de datos vuelve a estar disponible, los datos externos pasan a través de los *latches*, lo que da lugar al uso del término *latch* (*transparente*). El *latch* tipo D realiza esta función y cuando se activa, los datos que hay en su entrada aparecen en la salida del mismo modo que si se tratara de una conexión directa. Los datos de entrada se almacenan tan pronto como el *latch* se desactiva.

El latch S-R (SET-RESET)

Un **latch** es un tipo de dispositivo lógico **biestable** o **multivibrador**. Un latch S-R (Set-Reset) con entrada activa a nivel ALTO se compone de dos puertas NOR acopladas tal como se muestra en la Figura 8.1(a); un latch $\overline{S}$-$\overline{R}$ con entrada activa a nivel BAJO está formado por dos puertas NAND conectadas tal como se muestra en la Figura 8.1(b). Obsérvese que la salida de cada puerta se conecta a la entrada de la puerta opuesta. Esto origina la **realimentación** (feedback) regenerativa característica de todos los latches y flip-flops.



(a) Latch S-R con entrada activa a nivel ALTO

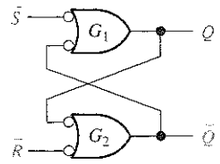
(b) Latch $\overline{S}$-$\overline{R}$ con entrada activa a nivel BAJO

FIGURA 8.1
Dos versiones del latch S-R (SET-RESET).

Para explicar el funcionamiento del latch, vamos a utilizar el latch $\overline{S}$-$\overline{R}$ de puertas NAND de la Figura 8.1(b). Este latch lo hemos vuelto a dibujar en la Figura 8.2, utilizando puertas negativa-OR equivalentes, debido a que los niveles BAJOS de las líneas $\overline{S}$ y $\overline{R}$ son las entradas de activación.

El latch de la Figura 8.2 tiene dos entradas, $\overline{S}$ y $\overline{R}$, y dos salidas Q y $\overline{Q}$. Asumimos que las dos entradas y la salida Q están a nivel ALTO. Dado que la salida Q se realimenta a una entrada de la puerta G_2 y que la entrada $\overline{R}$ está a nivel ALTO, la salida de G_2 tiene que ser un nivel BAJO. Esta salida a nivel BAJO está acoplada de nuevo a una entrada de la puerta G_1, asegurando así que su salida sea un nivel ALTO.

FIGURA 8.2
Equivalente con puertas negativa-OR para el latch S-R de puertas NAND de la Figura 8.1(b).



Cuando la salida Q está a nivel ALTO, el latch se encuentra en estado **SET** y permanecerá indefinidamente en él hasta que se aplique un nivel BAJO a la entrada $\overline{R}$. Si tenemos un nivel BAJO en la entrada $\overline{R}$ y un nivel ALTO en $\overline{S}$, la salida de la puerta G_2 se pone forzosamente a nivel ALTO. Este nivel ALTO en la salida $\overline{Q}$ se realimenta a una de las entradas de G_1 y, dado que la entrada $\overline{S}$ está a nivel ALTO, la salida de G_1 se pone a nivel BAJO. Este nivel BAJO en la salida Q se realimenta a una de las entradas de G_2, asegurando que la salida $\overline{Q}$ permanezca a nivel ALTO incluso cuando se elimine el nivel BAJO de la entrada $\overline{R}$. Cuando la salida Q es un nivel BAJO, el latch se encuentra en estado **RESET.** Ahora el latch permanece indefinidamente en este estado hasta que se aplique un nivel BAJO en la entrada $\overline{S}$.

En operación normal, las salidas de un latch son siempre complementarias una de la otra:

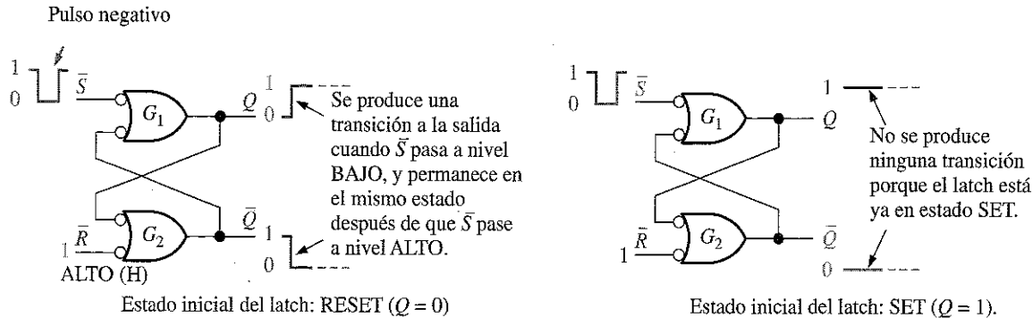
Cuando Q está a nivel ALTO, $\overline{Q}$ está a nivel BAJO y cuando Q está a nivel BAJO, $\overline{Q}$ está a nivel ALTO.

Se produce una condición de funcionamiento no válida en un latch $\overline{S}$-$\overline{R}$ con entradas activas a nivel BAJO, cuando se aplican simultáneamente niveles bajos a las dos entradas, $\overline{S}$ y $\overline{R}$. Mientras que se mantengan las dos entradas a nivel BAJO, las dos salidas deberían forzosamente estar a nivel ALTO, lo que viola la condición de complementariedad de las salidas. Además, si se eliminan simultáneamente los niveles bajos, las dos salidas van a tender al nivel BAJO y, dado que siempre va a existir un cierto retraso de propagación de la señal eléctrica a través de las puertas, una de las puertas dominará en la transición a nivel BAJO. Esto hará que la salida de la puerta más lenta permanezca a nivel ALTO. Cuando se produce esta situación, no se puede predecir el siguiente estado del latch.

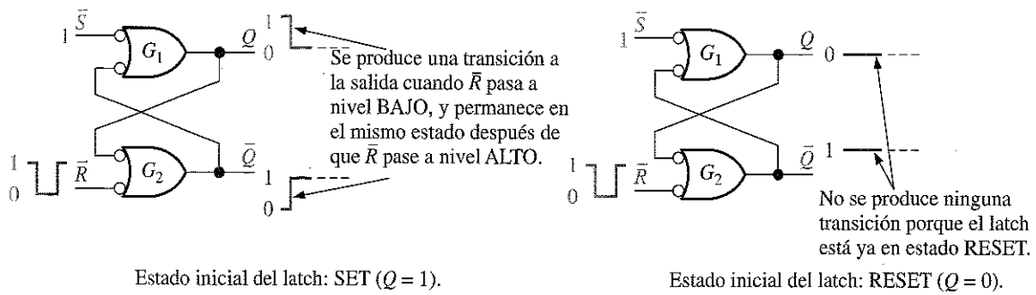
La Figura 8.3 ilustra el funcionamiento del latch $\overline{S}$-$\overline{R}$ con entradas activas a nivel BAJO, para cada una de las cuatro posibles combinaciones de los niveles de entrada. Las primeras tres combinaciones son válidas, no así la última. La Tabla 8.1 resume en forma de tabla de verdad el funcionamiento lógico. El funcionamiento del latch construido con puertas NOR con entradas activas a nivel ALTO de la Figura 8.1(a) es similar, pero requiere el uso de niveles lógicos opuestos.

Los símbolos lógicos para ambos tipos de latches, con entradas activas a nivel ALTO y a nivel BAJO, se muestran en la Figura 8.4.

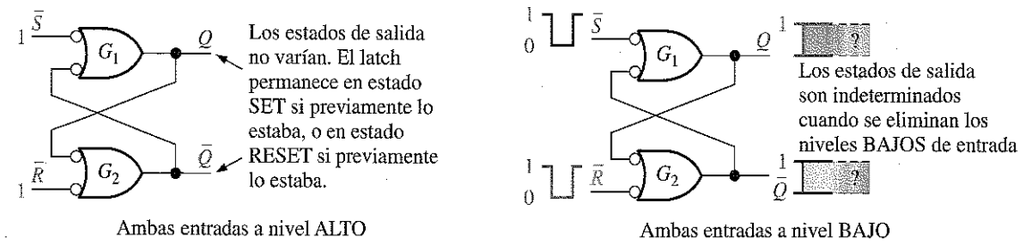
El Ejemplo 8.1 ilustra cómo un latch $\overline{S}$-$\overline{R}$ con entradas activas a nivel BAJO responde a las condiciones de entrada. Los niveles BAJOS se aplican a las entradas siguiendo una determinada secuencia y se observa la señal de salida Q resultante. La condición $\overline{S} = 0$ y $\overline{R} = 0$ no se contempla, ya que origina un modo de funcionamiento no válido del latch, lo que es un gran inconveniente en cualquier latch de tipo SET-RESET.



(a) Dos posibilidades para la operación SET



(b) Dos posibilidades para la operación RESET.



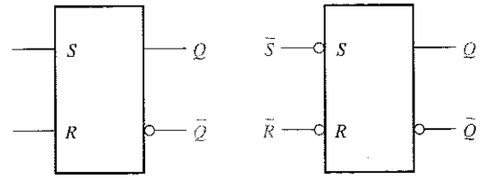
(c) Condición de no cambio

(d) Condición no válida

FIGURA 8.3
Los tres modos básicos de funcionamiento del latch $\overline{S}$-$\overline{R}$ (SET, RESET y no cambio) y la condición no válida.

TABLA 8.1
Tabla de verdad para un latch $\overline{S}$-$\overline{R}$ con entrada activa a nivel BAJO.

Entradas		Salidas		
$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$	Observaciones
1	1	NC	NC	No cambio. El latch permanece en el estado que estaba
0	1	1	0	Latch en estado SET.
1	0	0	1	Latch en estado RESET.
0	0	1	1	Condición no válida.



(a) Latch S-R con entrada activa a nivel ALTO

(b) Latch $\overline{S}$-$\overline{R}$ con entrada activa a nivel BAJO

FIGURA 8.4
Símbolos lógicos para los latches S-R y $\overline{S}$-$\overline{R}$.

EJEMPLO 8.1 Si se aplican las formas de onda $\overline{S}$ y $\overline{R}$ de la Figura 8.5(a) a las entradas del latch de la Figura 8.4(b), determinar la forma de onda que se observará en la salida Q. Suponer que Q se encuentra inicialmente a nivel BAJO.

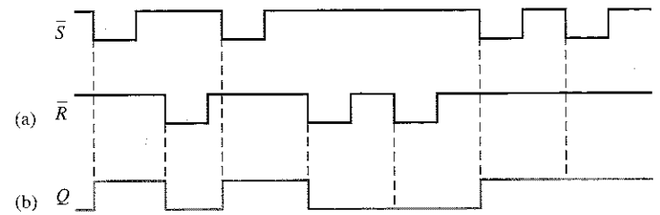


FIGURA 8.5

Solución. Véase la Figura 8.5(b)

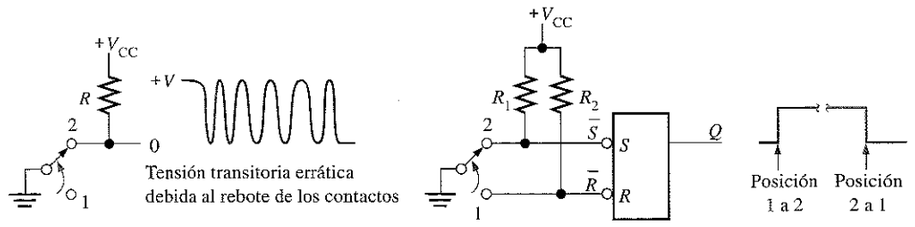
Problema relacionado. Determinar la salida Q de un latch S-R con entradas activas a nivel ALTO si se invierten las formas de onda de la Figura 8.5(a) y se aplican a las entradas.

Ejemplo de aplicación

El latch como eliminador del rebote de los contactos. Un buen ejemplo de aplicación de un latch $\overline{S}$-$\overline{R}$ consiste en la eliminación del "rebote" producido por los contactos de un interruptor mecánico. Cuando el polo de un interruptor choca con el contacto de cierre del interruptor, vibra o rebota varias veces hasta que, finalmente, se consigue un contacto firme. Aunque estos rebotes son mínimos, producen unos picos de tensión que pueden ser inadmisibles en un sistema digital. Esta situación se observa en la Figura 8.6(a).

Se puede utilizar un latch $\overline{S}$-$\overline{R}$ para eliminar los efectos de los rebotes del interruptor, como se muestra en la Figura 8.6(b). El interruptor se encuentra normalmente en la posición 1, manteniendo la entrada $\overline{R}$ a nivel BAJO y al latch en estado

RESET. Cuando el interruptor pasa a la posición 2, $\overline{R}$ pasa a nivel ALTO debido a la resistencia de pull-up conectada a V_{CC} y $\overline{S}$ pasa a nivel BAJO cuando se produce el primer contacto. Aunque $\overline{S}$ permanece a nivel BAJO durante un breve espacio de tiempo antes de que el interruptor rebote, este tiempo es suficiente para activar (SET) el latch. Cualquier otro pico de tensión aplicado posteriormente a la entrada $\overline{S}$, debido al rebote del interruptor, no va a afectar al latch, y éste permanecerá en el estado SET. Téngase en cuenta que la salida Q del latch proporciona una transición limpia del nivel BAJO al nivel ALTO, por lo que se eliminan los picos de tensión causados por el rebote de los contactos. De forma similar, se produce una transición limpia de nivel ALTO a nivel BAJO cuando el interruptor vuelve a la posición 1.

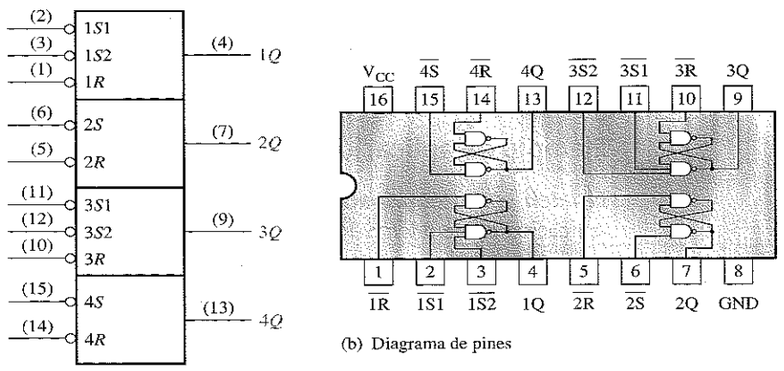


(a) Rebote del contacto de un interruptor (b) Circuito eliminador del rebote de los contactos

FIGURA 8.6
Utilización del latch $\overline{S}$-$\overline{R}$ para eliminar el rebote de los contactos de un interruptor.

Circuito integrado latch SET-RESET

El 74LS279 es un cuádruple latch $\overline{S}$-$\overline{R}$ representado por el diagrama lógico de la Figura 8.7(a) y cuyo diagrama de pines es el mostrado en la parte (b) de la misma figura. Obsérvese que dos de los latches tienen dos entradas $\overline{S}$.

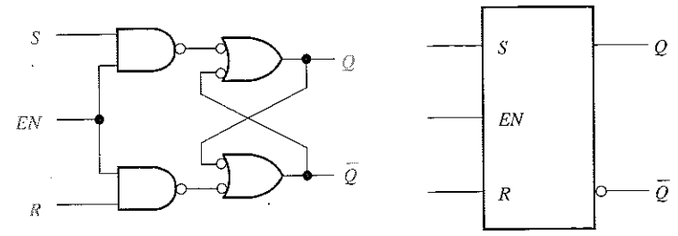


(a) Diagrama lógico (b) Diagrama de pines

FIGURA 8.7
El cuadrúple latch $\overline{S}$-$\overline{R}$ 74LS279.

El latch S-R con entrada de habilitación

El diagrama y el símbolo lógico de un latch con entrada de habilitación se muestran en la Figura 8.8. Las entradas S y R controlan el estado al que va a cambiar el latch cuando se aplica un nivel ALTO a la entrada de habilitación (*EN, enable*). El latch no cambia de estado hasta que la entrada *EN* está a nivel ALTO pero, mientras que permanezca en este estado, la salida va a ser controlada por el estado de las entradas S y R. En este circuito, el estado no válido del latch se produce cuando las dos entradas S y R están simultáneamente a nivel ALTO.



(a) Diagrama lógico (b) Símbolo lógico

FIGURA 8.8
Latch S-R con entrada de habilitación.

EJEMPLO 8.2 Determinar la forma de onda de salida Q si se aplican las señales de entrada mostradas en la Figura 8.9(a) a un latch S-R con entrada de habilitación, que se encuentra inicialmente en estado de RESET.

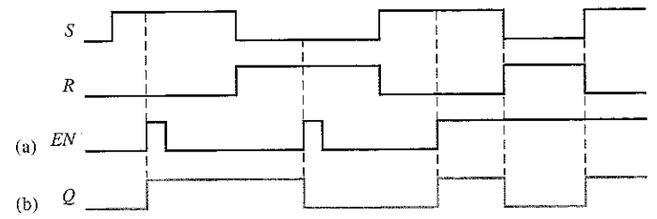


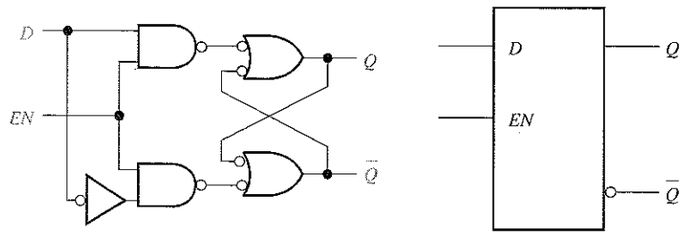
FIGURA 8.9

Solución. La forma de onda Q se muestra en la Figura 8.9(b). Siempre que S está a nivel ALTO y R a nivel BAJO, un nivel ALTO en la entrada *EN* hace que el latch se ponga en estado SET. Siempre que S está a nivel BAJO y R a nivel ALTO, un nivel ALTO en la entrada *EN* hace que el latch se ponga en estado RESET.

Problema relacionado. Determinar la salida Q de un latch S-R con entrada de habilitación, si se invierten las entradas S y R de la Figura 8.9(a).

El latch D con entrada de habilitación

Existe otro tipo de latch con entrada de habilitación que se denomina latch D. Se diferencia del latch S-R en que sólo tiene una entrada, además de la de habilitación, *EN*. Esta entrada recibe el nombre de entrada de datos (*D*). La Figura 8.10 muestra el diagrama y el símbolo lógico de este tipo de latch. Cuando la entrada *D* está a nivel ALTO y la entrada *EN* también, el latch se pone en estado SET. Cuando la entrada *D* está a nivel BAJO y la entrada *EN* está a nivel ALTO, el latch se pone en estado RESET. Dicho de otra manera, la salida *Q* es igual a la entrada *D* cuando la entrada de habilitación *EN* está a nivel ALTO.

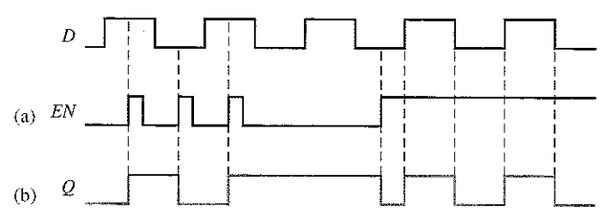


(a) Diagrama lógico (b) Símbolo lógico

FIGURA 8.10
Latch D con entrada de habilitación.

EJEMPLO 8.3 Determinar la forma de onda de salida *Q* si se aplican las entradas que se muestran en la Figura 8.11(a) a un latch D con entrada de habilitación que, inicialmente, está en estado RESET.

FIGURA 8.11

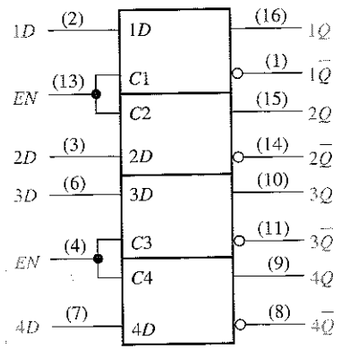


Solución. La forma de onda *Q* se muestra en la Figura 8.11(b). Siempre que *D* y *EN* estén a nivel ALTO, la salida *Q* será un nivel ALTO. Siempre que *D* sea un nivel BAJO y *EN* esté a nivel ALTO, *Q* se pondrá a nivel BAJO. Cuando *EN* está a nivel BAJO, el estado del latch no se ve afectado por la entrada *D*.

Problema relacionado. Determinar la salida *Q* del latch D con entrada de habilitación, si la señal de entrada *D* de la Figura 8.11(a) se invierte.

Circuito integrado latch D

Un ejemplo de un CI latch D con entrada de habilitación es el 74LS75, cuyo símbolo lógico se puede ver en la Figura 8.12(a). Este dispositivo está compuesto por cuatro latches. Observe que cada entrada de habilitación *EN* activa a nivel ALTO está compartida por dos latches y se designa como entrada de control (*C*). La tabla de verdad de cada latch se muestra en la Figura 8.12(b). La X en esta tabla representa una condición "indiferente". En este caso, cuando la entrada *EN* está a nivel BAJO, da lo mismo el valor que tenga la entrada *D*, ya que las salidas no se ven afectadas y permanecen en los estados en que se encontraban.



(a) Símbolo lógico

Entradas		Salidas		Observaciones
D	*EN*	*Q*	$\overline{Q}$	
0	1	0	1	RESET
1	1	1	0	SET
X	0	Q_0	$\overline{Q}_0$	No cambio

NOTA: Q_0 es el nivel de salida previo, antes de que se establecieran las condiciones de entrada indicadas

(b) Tabla de verdad (de cada latch)

FIGURA 8.12
El cuádruple latch D con entrada de habilitación 74LS75.

REVISIÓN DE LA SECCIÓN 8.1

1. Enumerar tres tipos de latches.
2. Desarrollar la tabla de verdad del latch S-R con entradas activas a nivel ALTO de la Figura 8.1(a).
3. ¿Cuál es la salida *Q* de un latch D cuando *EN* = 1 y *D* = 1?

8.2 ■ FLIP-FLOPS DISPARADOS POR FLANCO

Los flip-flops son dispositivos síncronos de dos estados, también conocidos como multivibradores biestables. En este caso, el término síncrono significa que la salida cambia de estado únicamente en un instante específico de una entrada de disparo denominada reloj (CLK), la cual recibe el nombre de entrada de control, C. Esto significa que los cambios en la salida se producen sincronizadamente con el reloj.

Al finalizar esta sección, el lector deberá ser capaz de:

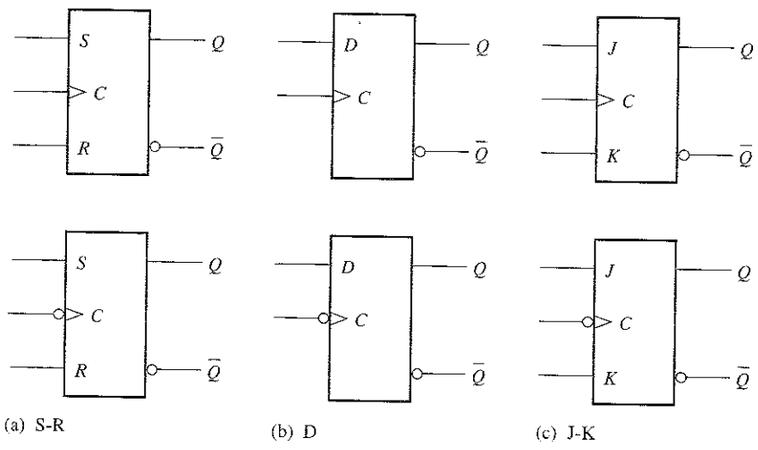
■ Definir *reloj*. ■ Definir *flip-flop disparado por flanco*. ■ Explicar la diferencia entre un flip-flop y un latch. ■ Identificar un flip-flop disparado por flanco mediante su

símbolo lógico. ■ Comentar la diferencia entre los flip-flops disparados por flancos positivos y negativos. ■ Comparar el funcionamiento de los flip-flops disparados por flanco S-R, D y J-K, y explicar las diferencias entre sus tablas de verdad. ■ Explicar las entradas asíncronas de un flip-flop. ■ Describir los flip-flops 74AHC74 y 74HC112.

NOTAS INFORMÁTICAS

Las memorias semiconductoras para computadoras constan de numerosas celdas individuales. Cada celda de almacenamiento contiene un 1 o un 0. Un tipo de memoria es la memoria de acceso aleatorio estática o SRAM, que utiliza flip-flops como celdas de almacenamiento, ya que un flip-flop mantendrá uno de dos estados de forma indefinida siempre que se aplique alimentación continua; de aquí el término *estática*. Este tipo de memoria se clasifica como una memoria *volátil*, dado que todos los datos almacenados se perderán cuando se desconecte la alimentación. Existe otro tipo de memoria, la memoria RAM dinámica o DRAM, que utiliza capacitancias en lugar de flip-flops como elemento básico de almacenamiento y debe refrescarse periódicamente para mantener los datos almacenados.

Un **flip-flop disparado por flanco** cambia de estado con el flanco positivo (flanco de subida) o con el flanco negativo (flanco de bajada) del impulso de reloj y es sensible a sus entradas sólo en esta transición del reloj. En esta sección se cubren tres tipos de flip-flops disparados por flanco: S-R, D y J-K. Los símbolos lógicos de estos dispositivos se muestran en la Figura 8.13. Obsérvese que pueden ser disparados por flanco positivo (no hay círculo en la entrada *C*) o por flanco negativo (hay un círculo en la entrada *C*). La clave para identificar un flip-flop disparado por flanco mediante su símbolo lógico la da el triángulo que se encuentra dentro del bloque en la entrada del reloj (*C*). El triángulo se denomina *indicador de entrada dinámica*.



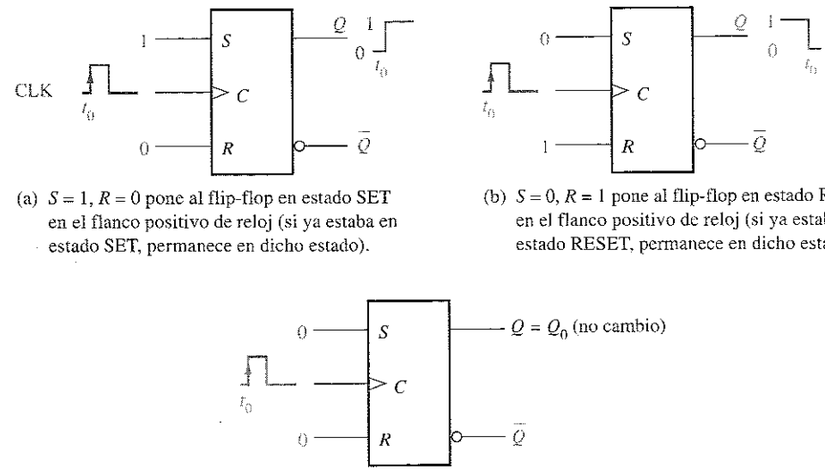
(a) S-R (b) D (c) J-K

FIGURA 8.13
Símbolos lógicos de los latches disparados por flanco (parte superior: disparado por flanco positivo; parte inferior: disparado por flanco negativo).

Flip-flop S-R disparado por flanco

Las entradas *S* y *R* de un **flip-flop S-R** se denominan entradas **síncronas**, dado que los datos en estas entradas se transfieren a las salidas del flip-flop sólo con el flanco de disparo del impulso del reloj. Cuando *S* está a nivel ALTO y *R* está a nivel BAJO, la salida *Q* se pone a nivel ALTO con el flanco de disparo del impulso de reloj, pasando el flip-flop al estado SET. Cuando *S* está a nivel BAJO y *R* está a nivel ALTO, la salida *Q* se pone a nivel BAJO con el flanco de disparo del impulso de reloj, pasando el flip-flop al estado RESET. Cuando tanto *S* como *R* están a nivel BAJO, la salida no cambia de estado. Cuando *S* y *R* están a nivel ALTO, se produce una condición no válida.

El funcionamiento básico de un flip-flop disparado por flanco positivo se muestra en la Figura 8.14, mientras que la tabla de verdad se puede ver en la Tabla 8.2. Recordemos que *un flip-flop no puede cambiar de estado excepto en el flanco de disparo de un impulso de reloj*. Las entradas *S* y *R* se pueden cambiar en cualquier instante en que la entrada de reloj esté a nivel ALTO o nivel BAJO (excepto durante un breve instante de tiempo en las proximidades de las transiciones de disparo del reloj) sin que varíe la salida.



(a) $S = 1$, $R = 0$ pone al flip-flop en estado SET en el flanco positivo de reloj (si ya estaba en estado SET, permanece en dicho estado).

(b) $S = 0$, $R = 1$ pone al flip-flop en estado RESET en el flanco positivo de reloj (si ya estaba en estado RESET, permanece en dicho estado).

(c) $S = 0$, $R = 0$ no varía el estado en que se encuentre el flip-flop (si está en estado SET permanece en este estado; si está en estado RESET permanece en dicho estado).

FIGURA 8.14
Funcionamiento de un flip-flop S-R disparado por flanco positivo.

TABLA 8.2
Tabla de verdad de un flip-flop S-R disparado por flanco positivo.

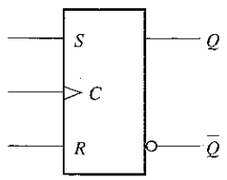
Entradas			Salidas		
S	*R*	CLK	*Q*	$\overline{Q}$	Observaciones
0	0	X	Q_0	$\overline{Q}_0$	No cambio
0	1	↑	0	1	RESET
1	0	↑	1	0	SET
1	1	↑	?	?	No válido

↑ = transición del reloj de nivel bajo (L) a nivel alto (H)
X = irrelevante (condición indiferente)
Q_0 = nivel de salida previo a la transición del reloj

El funcionamiento y tabla de verdad de un flip-flop S-R disparado por flanco negativo son los mismos que los de un dispositivo disparado por flanco positivo, excepto en que el flanco de bajada del impulso del reloj es, en este caso, el flanco de disparo.

EJEMPLO 8.4 Determinar las formas de onda de salida Q y $\bar{Q}$ del flip-flop de la Figura 8.15, para las entradas *S, R* y *CLK* de la Figura 8.16(a). Suponer que el flip-flop disparado por flanco positivo se encuentra, inicialmente, en estado RESET.

FIGURA 8.15



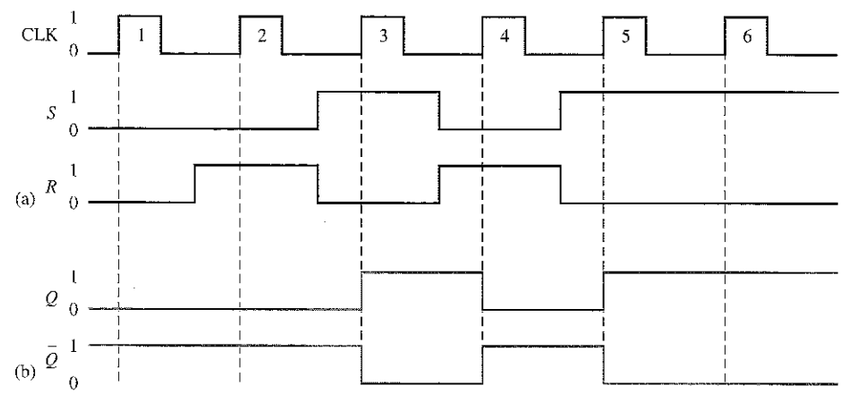


FIGURA 8.16

Solución

1. Durante el impulso 1 de reloj, *S* está a nivel BAJO y *R* está a nivel BAJO, luego Q no cambia.
2. Durante el impulso 2 de reloj, *S* está a nivel BAJO y *R* está a nivel ALTO, luego Q permanece a nivel BAJO (RESET).
3. Durante el impulso 3 de reloj, *S* está a nivel ALTO y *R* está a nivel BAJO, luego Q pasa a nivel ALTO (SET).
4. Durante el impulso 4 de reloj, *S* está a nivel BAJO y *R* está a nivel ALTO, luego Q pasa a nivel BAJO (RESET).
5. Durante el impulso 5 de reloj, *S* está a nivel ALTO y *R* está a nivel BAJO, luego Q pasa a nivel ALTO (SET).
6. Durante el impulso 6 de reloj, *S* está a nivel ALTO y *R* está a nivel BAJO, luego Q permanece a nivel ALTO.

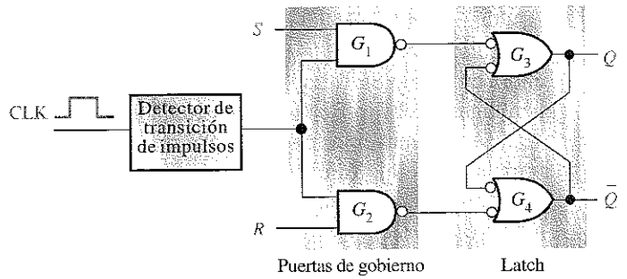
Una vez que se ha determinado Q, se puede conocer $\bar{Q}$ de forma muy sencilla, complementando la salida Q. Las formas de onda resultantes para Q y $\bar{Q}$ se muestran en la Figura 8.16(b) en función de las formas de onda de entrada de la parte (a).

Problema relacionado. Determinar Q y $\bar{Q}$ para las entradas *S* y *R* de la Figura 8.16(a), si el flip-flop es disparado por flanco negativo.

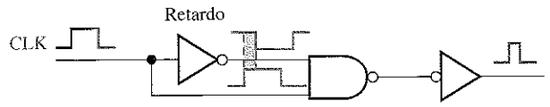
NOTAS INFORMÁTICAS Todas las operaciones lógicas que se realizan mediante hardware también pueden implementarse por software. Por ejemplo, la operación de un flip-flop J-K puede realizarse mediante instrucciones específicas de computadora. Si se utilizaran dos bits para representar las entradas J y K, la computadora no haría nada para la entrada 00; para la entrada 10, un bit de datos que representara la salida Q se pondría a 1, el bit de datos Q sería 0 para la entrada 01 y el bit de datos Q se complementaría para la entrada 11. Aunque no suele ser habitual utilizar una computadora para simular un flip-flop, la cuestión es que todas las operaciones de hardware se pueden realizar mediante software.

Un método de disparo por flanco

En la Figura 8.17(a) se presenta la implementación simplificada de un flip-flop S-R disparado por flanco, que se va a utilizar para explicar el concepto de disparo por flanco, aunque esto no significa que sea el tipo de flip-flop más importante. Realmente, los flip-flops D y J-K se utilizan con más frecuencia, y se pueden encontrar como circuitos integrados mucho más fácilmente que los S-R. Sin embargo, es mucho más interesante el estudio de los S-R, ya que tanto el flip-flop J-K como el D se pueden derivar de él. Conviene tener en cuenta que el flip-flop S-R se distingue del latch S-R con entrada de habilitación únicamente en que cuenta con un detector de transiciones de impulsos. Este circuito produce un pico de muy corta duración en la transición positiva del impulso de reloj.



(a) Diagrama lógico simplificado de un flip-flop S-R disparado por flanco positivo



(b) Un tipo de detector de transiciones de impulsos

FIGURA 8.17
Disparo por flanco.

Un detector de transiciones de impulsos típico se muestra en la Figura 8.17(b). Como se puede ver, existe un pequeño retraso en una de las entradas de la puerta NAND de manera que el impulso invertido de reloj llega a la entrada de la puerta unos cuantos nanosegundos después que el verdadero impulso de reloj. Esto origina un pico de salida que dura sólo unos nanosegundos. En los flip-flops disparados por flanco negativo, se invierte primero el impulso de reloj, de forma que se origina un pico muy estrecho en el flanco de bajada.

Obsérvese que el circuito de la Figura 8.17 está dividido en dos secciones, una correspondiente a las denominadas puertas de gobierno, y otra al latch. Las puertas de gobierno dirigen los picos de reloj hacia la entrada de la puerta G_3 o la entrada de la G_4, dependiendo del estado de *S* y *R*. Para comprender el funcionamiento de este flip-flop, vamos a comenzar suponiendo que se encuentra en estado RESET ($Q = 0$) y que las entradas *S, R* y *CLK* están todas a nivel BAJO. En esta situación, las salidas de las

puertas G_1 y G_2 están ambas a nivel ALTO. La salida Q a nivel BAJO se realimenta a una de las entradas de la puerta G_4, forzando la salida $\overline{Q}$ a nivel ALTO. Puesto que $\overline{Q}$ está a nivel ALTO, las dos entradas de la puerta G_3 están a nivel ALTO (recordemos que la salida de la puerta G_1 está a nivel ALTO), manteniendo la salida Q a nivel BAJO. Si se aplica un impulso a la entrada de reloj CLK, las salidas de las puertas G_1 y G_2 permanecen a nivel ALTO, ya que se desactivan cuando las entradas S y R están a nivel BAJO; por consiguiente, no hay ningún cambio en el estado del flip-flop: permanece en RESET.

Ahora, se pone la entrada S a nivel ALTO, dejando R a nivel BAJO y se aplica un impulso de reloj. Dado que la entrada S de la puerta G_1 está ahora a nivel ALTO, la salida de la puerta G_1 pasa a nivel BAJO durante un breve espacio de tiempo (pico) cuando CLK pasa a nivel ALTO, haciendo que la salida Q se ponga a nivel ALTO. Las dos entradas de la puerta G_4 están ahora a nivel ALTO (recordemos que la salida de G_2 está a nivel ALTO ya que R está a nivel BAJO), forzando la salida $\overline{Q}$ a pasar a nivel BAJO. Este nivel BAJO de la salida $\overline{Q}$ se realimenta a una de las entradas de la puerta G_3, asegurando que la salida Q permanezca a nivel ALTO. El flip-flop se encuentra ahora en estado SET. La Figura 8.18 ilustra las transiciones de niveles lógicos que tienen lugar en el flip-flop para esta condición.

A continuación, ponemos S a nivel BAJO y R a nivel ALTO, y aplicamos un impulso de reloj. Ya que la entrada R está a nivel ALTO, el flanco positivo de reloj produce un pico negativo en la salida de la puerta G_2, haciendo que la salida $\overline{Q}$ pase a nivel ALTO. Debido a este nivel ALTO en $\overline{Q}$, ambas entradas de la puerta G_3 están ahora a nivel ALTO (recordemos que la salida de la puerta G_1 es un nivel ALTO debido a que la entrada S está a nivel BAJO), forzando a la salida Q a pasar a nivel BAJO. Este nivel BAJO en Q se realimenta a una de las entradas de la puerta G_4, asegurando así que $\overline{Q}$ permanecerá a nivel ALTO. El flip-flop se encuentra ahora en estado RESET. La Figura 8.19 ilustra las transiciones de niveles lógicos que ocurren en el flip-flop para esta condición. Al igual que en el latch con entrada de habilitación, se produce una condición no válida cuando ambas entradas S y R están, simultáneamente, a nivel ALTO. Esta es la principal desventaja de los flip-flops S-R.

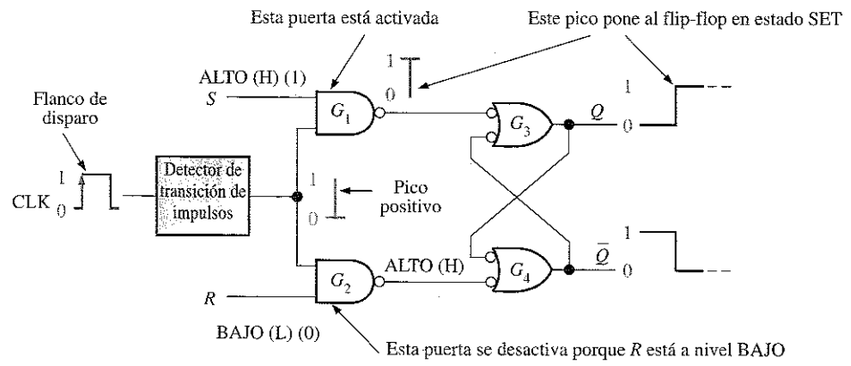


FIGURA 8.18
Flip-flop que realiza una transición del estado RESET al estado SET durante el flanco positivo del impulso del reloj.

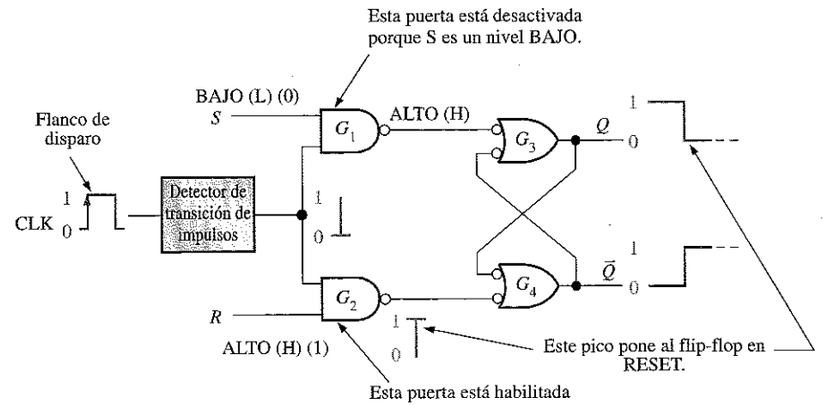


FIGURA 8.19
Flip-flop que realiza una transición del estado SET al estado RESET durante el flanco positivo del impulso del reloj.

El flip-flop D disparado por flanco

El **flip-flop D** resulta muy útil cuando se necesita almacenar un único bit de datos (1 o 0). Si se añade un inversor a un flip-flop S-R obtenemos un flip-flop D básico, como se muestra en la Figura 8.20, en la que se muestra uno disparado por flanco positivo.

Obsérvese que el flip-flop de la Figura 8.20 tiene únicamente una entrada, la entrada D, además del reloj. Si cuando se aplica un impulso de reloj la entrada D está a nivel ALTO, el flip-flop se activa (SET) y almacena el nivel ALTO de la entrada D durante el flanco positivo del impulso del reloj. Si existe un nivel BAJO en la entrada D cuando se aplica el impulso del reloj, el flip-flop se pone a cero (RESET) y almacena el nivel BAJO de la entrada D durante el flanco de bajada del impulso del reloj. En el estado SET, el flip-flop almacena un 1, mientras que en el estado RESET almacena un 0.

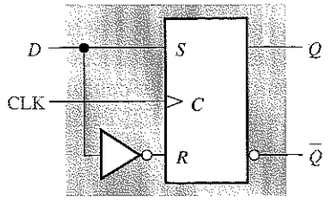


FIGURA 8.20
Flip-flop D disparado por flanco positivo, formado por un flip-flop S-R y un inversor.

El funcionamiento de un flip-flop D disparado por flanco positivo se resume en la Tabla 8.3. El funcionamiento de un dispositivo activado por flanco negativo es, por supuesto, idéntico, excepto que el disparo tiene lugar en el flanco de bajada del impulso del reloj. *Recuérdese que Q sigue a D en cada flanco del impulso de reloj.*

TABLA 8.3
Tabla de verdad de un flip-flop D disparado por flanco positivo.

Entradas		Salidas		
D	CLK	Q	$\overline{Q}$	Observaciones
1	↑	1	0	SET (almacena un 1)
0	↑	0	1	RESET (almacena un 0)

↑ = transición del reloj de nivel BAJO (L) a ALTO (H)

EJEMPLO 8.5 Dadas las formas de onda de la Figura 8.21(a) para la entrada D y el reloj, determinar la onda de salida Q si el flip-flop parte del estado RESET.

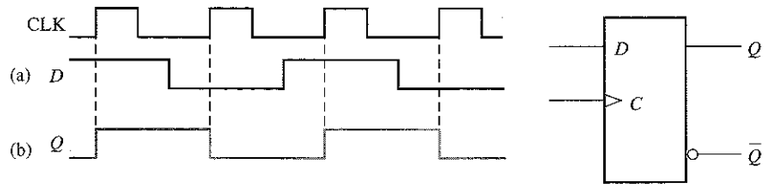


FIGURA 8.21

Solución. La salida Q sigue al estado de la entrada D cada vez que se produce un flanco positivo del reloj. La salida resultante se muestra en la Figura 8.21(b).

Problema relacionado. Determinar la salida Q para el flip-flop D, si la entrada D de la Figura 8.21(a) se invierte.

El flip-flop J-K disparado por flanco

El **flip-flop J-K** es versátil y es uno de los tipos de flip-flop más ampliamente utilizado. Las denominaciones J y K de sus entradas no tienen ningún significado conocido, excepto el hecho de que son dos letras consecutivas del alfabeto.

El funcionamiento del flip-flop J-K es idéntico al del flip-flop S-R en las condiciones de operación SET, RESET y de permanencia de estado. *La diferencia está en que el flip-flop J-K no tiene condiciones no válidas como ocurre en el S-R.*

La Figura 8.22 muestra la lógica interna de un flip-flop J-K disparado por flanco positivo. Obsérvese que se diferencia del flip-flop S-R disparado por flanco en que la salida Q se realimenta a la entrada de la puerta G_2, y la salida $\overline{Q}$ se realimenta a la entrada de la puerta G_1. Las dos entradas se denominan J y K. Un flip-flop J-K puede ser también del tipo disparado por flanco negativo, en cuyo caso, la entrada de reloj se invierte.

Supongamos que el flip-flop de la Figura 8.23 se encuentra en estado RESET y que la entrada J está a nivel ALTO y la entrada K está a nivel BAJO. Cuando se produce un impulso de reloj, pasa un pico correspondiente al flanco anterior, indicado por ①, a través de la puerta G_1, ya que $\overline{Q}$ está a nivel ALTO y J también está a nivel ALTO. Esto origina que la parte latch del flip-flop cambie al estado SET.

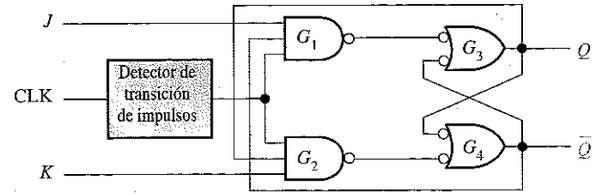


FIGURA 8.22
Diagrama lógico simplificado de un flip-flop J-K disparado por flanco positivo.

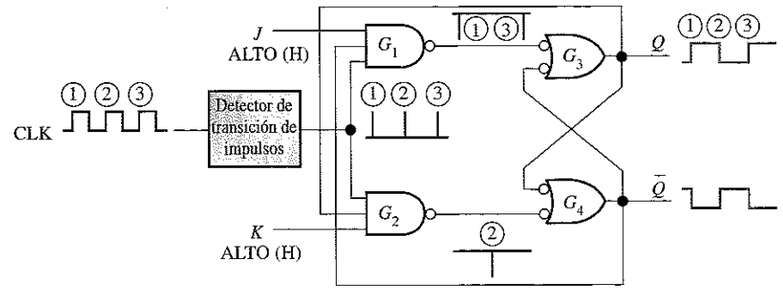


FIGURA 8.23
Transiciones que ilustran el modo de basculación cuando J = 1 y K = 1.

El flip-flop se encuentra ahora en estado SET. Si ponemos la entrada J a nivel BAJO y la entrada K a nivel ALTO, el siguiente pico del reloj, indicado por ②, pasará a través de la puerta G_2, ya que Q está a nivel ALTO y K también. Esto origina que la parte latch del flip-flop cambie al estado RESET.

Si ahora se aplica un nivel BAJO a las dos entradas J y K, el flip-flop permanecerá en su estado actual cuando se produzca un impulso del reloj. De esta manera, un nivel BAJO en J y K origina una condición *de no cambio.*

Hasta ahora, el funcionamiento lógico del flip-flop J-K es idéntico al del S-R en sus modos SET, RESET y de no cambio. La diferencia tiene lugar cuando las dos entradas, J y K, están a nivel ALTO. Para ver esto, supongamos que el flip-flop se encuentra en estado RESET. El nivel ALTO de la salida $\overline{Q}$ activa la puerta G_1 de forma que el pico del reloj, indicado por ③, pasa y activa (SET) el flip-flop. Ahora hay un nivel ALTO en Q, el cual permite que el siguiente pico del reloj pase a través de la puerta G_2 y ponga el flip-flop en estado RESET.

Como puede verse, en cada pico sucesivo de reloj, el flip-flop cambia a su estado opuesto. A este modo de funcionamiento se le denomina **modo de basculación (toggle)**. La Figura 8.23 ilustra las transiciones cuando el flip-flop se encuentra en este modo.

En la Tabla 8.4 se muestra la tabla de verdad del flip-flop J-K disparado por flanco, la cual resume su funcionamiento. *Obsérvese que no hay ningún estado no válido, como ocurría con el flip-flop S-R.* La tabla de verdad de un dispositivo disparado por flanco negativo es idéntica, excepto en que se dispara durante el flanco de bajada del impulso de reloj.

TABLA 8.4
Tabla de verdad de un flip-flop J-K disparado por flanco positivo.

Entradas			Salidas		Observaciones
J	K	CLK	Q	$\overline{Q}$	
0	0	↑	Q_0	$\overline{Q}_0$	No cambio
0	1	↑	0	1	RESET
1	0	↑	1	0	SET
1	1	↑	$\overline{Q}_0$	Q_0	Basculación

↑ = transición del reloj del nivel BAJO al nivel ALTO

Q_0 = nivel de salida previo a la transición del reloj

EJEMPLO 8.6 Las formas de onda de entrada de la Figura 8.24(a) se aplican a las entradas *J*, *K* y de reloj, tal y como se muestra. Determinar la salida *Q* suponiendo que el flip-flop se encuentra inicialmente en estado RESET.

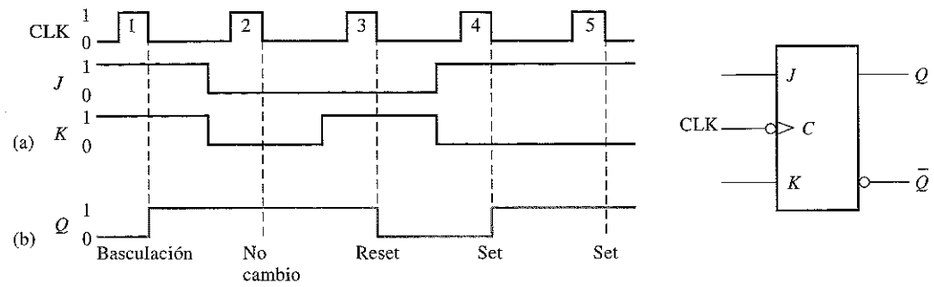


FIGURA 8.24

Solución

1. En primer lugar, dado que se trata de un flip-flop disparado por flanco negativo, como se indica mediante el círculo en la entrada de reloj, la salida *Q* cambiará sólo al ocurrir el flanco negativo del impulso de reloj.
2. En el primer impulso de reloj, *J* y *K* están a nivel ALTO y, debido a la condición de basculación, *Q* pasa a nivel ALTO.
3. En el segundo impulso de reloj, se produce la condición de no cambio en las entradas, manteniendo *Q* a nivel ALTO.
4. En el tercer impulso del reloj, *J* está a nivel BAJO y *K* a nivel ALTO, produciendo una condición de RESET, por lo que *Q* pasa a nivel BAJO.
5. En el cuarto impulso de reloj, *J* está a nivel ALTO y *K* a nivel BAJO, dando lugar a una condición de SET, luego *Q* pasa a nivel ALTO.
6. La condición SET permanece en *J* y *K* cuando ocurre el quinto impulso del reloj, de forma que *Q* sigue a nivel ALTO.

La forma de onda *Q* resultante se indica en la Figura 8.24(b).

Problema relacionado. Determinar la salida *Q* del flip-flop J-K si las entradas *J* y *K* de la Figura 8.24(a) se invierten.

EJEMPLO 8.7 Las formas de onda de la Figura 8.25(a) se aplican al flip-flop que se muestra. Determinar la salida *Q*, comenzando en el estado RESET.

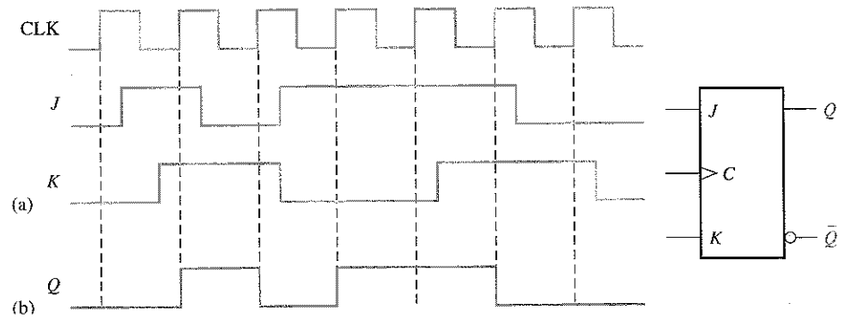


FIGURA 8.25

Solución. La salida *Q* toma el estado determinado por los estados de las entradas *J* y *K* en el flanco positivo (flanco de disparo) del impulso de reloj. Si se produce un cambio en *J* o en *K* después del disparo del reloj éste no tiene efecto en la salida, como se muestra en la Figura 8.25(b).

Problema relacionado. Intercambiar las entradas *J* y *K* y determinar la salida *Q* resultante.

Entradas asíncronas de inicialización y borrado

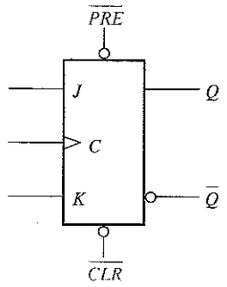
En los flip-flops que acabamos de estudiar, el *S-R*, el *D* y el *J-K*, se dice que sus entradas son *entradas síncronas,* ya que los datos de estas entradas condicionan la salida de los flip-flops sólo durante el flanco de disparo del impulso de reloj; esto significa que los datos se transfieren sincronizados con la señal de reloj.

La mayoría de los circuitos integrados flip-flops tienen también entradas **asíncronas**. Estas son entradas que pueden variar el estado del flip-flop *independientemente del reloj*. Generalmente, reciben el nombre de **inicialización —preset—** (*PRE*) y **borrado —clear—** (*CLR*), aunque también algunos fabricantes las designan como *activación directa* (S_D, direct SET) y *desactivación directa* (R_D, direct RESET). Un nivel activo en la entrada de inicialización del flip-flop (preset) pone a SET el dispositivo, y un nivel activo en la entrada de borrado (clear) lo pone en estado RESET. En la Figura 8.26 se muestra el símbolo lógico de un flip-flop J-K con entradas preset y clear. Estas entradas son activas a nivel BAJO, como indican los círculos. Estas entradas de inicialización y borrado deben mantenerse a nivel ALTO para el funcionamiento síncrono.

La Figura 8.27 muestra el diagrama lógico de un flip-flop J-K disparado por flanco con entradas de inicialización y de borrado activas a nivel BAJO ($\overline{PRE}$) y ($\overline{CLR}$). Esta figura ilustra, básicamente, cómo funcionan estas entradas. Como puede

verse, están conectadas de forma que anulan el efecto de las entradas síncronas J, K y el reloj.

FIGURA 8.26
Símbolo lógico de un flip-flop J-K con entrada de inicialización (preset) y de borrado (clear) activas a nivel BAJO.



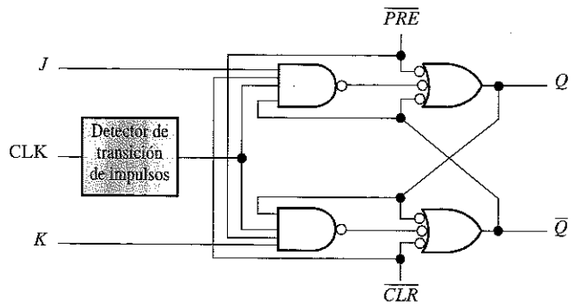


FIGURA 8.27
Diagrama lógico de un flip-flop J-K con entradas de inicialización (preset) y de borrado (clear) activas a nivel BAJO.

EJEMPLO 8.8 En el flip-flop J-K activado por flanco positivo de la Figura 8.28(a), con entradas preset y clear, determinar la salida Q para las entradas mostradas en el diagrama de tiempos, si Q está inicialmente a nivel BAJO.

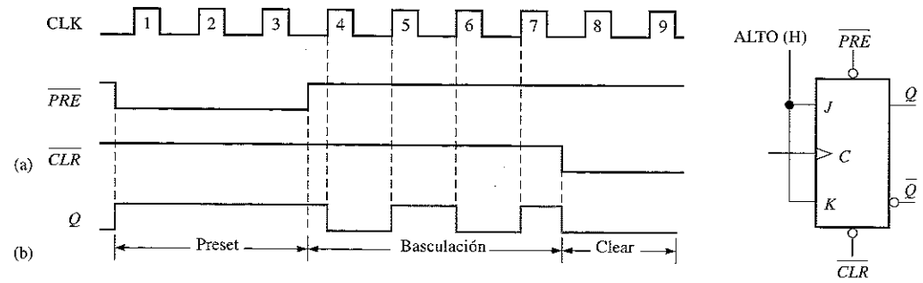


FIGURA 8.28

Solución

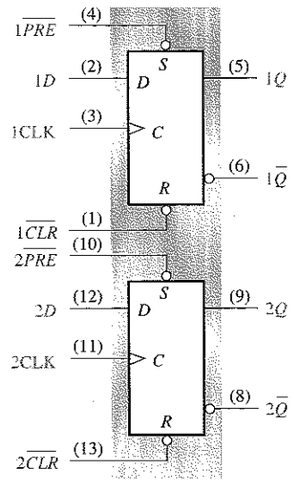
1. Durante los impulsos de reloj 1, 2 y 3, la entrada de inicialización ($\overline{PRE}$) está a nivel BAJO, manteniendo el flip-flop en estado SET, independientemente de las entradas síncronas J y K.
2. Durante los impulsos 4, 5, 6 y 7, funciona en modo de basculación, dado que J está a nivel ALTO, K está a nivel ALTO y tanto $\overline{PRE}$ como $\overline{CLR}$ están a nivel ALTO.
3. Para los impulsos de reloj 8 y 9, la entrada de borrado ($\overline{CLR}$) está a nivel BAJO, por lo que el flip-flop se mantiene en estado RESET, independientemente de las entradas síncronas. La salida Q resultante se muestra en la Figura 8.28(b).

Problema relacionado. Si intercambiamos las formas de onda de entrada $\overline{PRE}$ y $\overline{CLR}$ de la Figura 8.28(a), ¿qué forma tendrá la onda de salida Q?

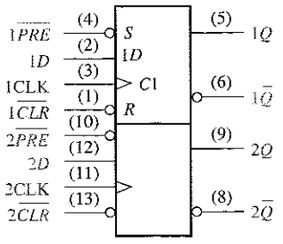
Circuitos integrados flip-flop

Ahora, se van a tratar dos circuitos integrados flip-flops disparados por flanco, que son representativos de varios tipos de flip-flops disponibles en forma de CI y que, al igual que la mayoría de otros dispositivos, se encuentran disponibles en TTL y CMOS.

Doble flip-flop D 74AHC74. El dispositivo CMOS contiene dos flip-flops idénticos que son independientes entre sí, excepto en que comparten V_{CC} y tierra. Son flip-flops disparados por flanco positivo y disponen de las entradas asíncronas de inicialización y borrado activas a nivel BAJO. En la Figura 8.29(a) se muestran los símbolos lógicos de cada flip-flop individual dentro del encapsulado, mientras que en la parte (b) de la figura podemos ver el símbolo estándar ANSI/IEEE, que representa el dispositivo completo. La numeración de los pines se indica entre paréntesis.



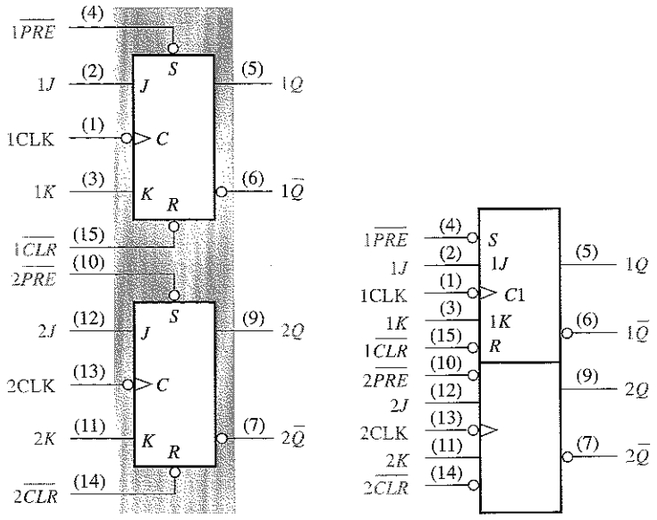
(a) Símbolos lógicos individuales



(b) Bloque lógico.
Nota: La S y la R dentro del bloque indican que $\overline{PRE}$ pone al dispositivo en estado SET y $\overline{CLR}$ le pone en estado RESET.

FIGURA 8.29
Símbolos lógicos del doble flip-flop D disparado por flanco positivo 74AHC74.

Doble flip-flop J-K 74HC112. Este dispositivo CMOS contiene también dos flip-flops idénticos que son disparados por flanco negativo, y tienen entradas asíncronas de inicialización y de borrado activas a nivel BAJO. Los símbolos lógicos correspondientes se muestran en la Figura 8.30.



(a) Símbolos lógicos individuales (b) Bloque lógico

FIGURA 8.30
Símbolos lógicos del doble flip-flop J-K disparado por flanco negativo 74HC112.

EJEMPLO 8.9 Las formas de onda para 1J, 1K, 1CLK, 1 $\overline{PRE}$ y 1 $\overline{CLR}$ de la Figura 8.31(a) se aplican a uno de los flip-flops disparados por flanco negativo del circuito 74HC112. Determinar la onda de salida 1Q.

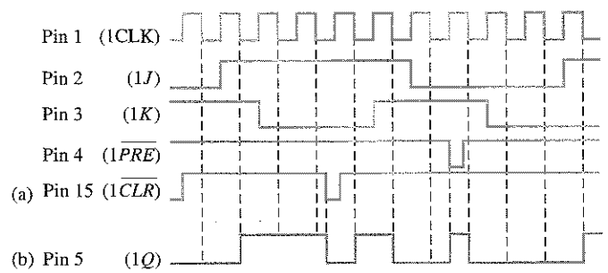


FIGURA 8.31

Solución. La forma de onda resultante 1Q se muestra en la Figura 8.31(b). Obsérvese que cada vez que se aplica un nivel BAJO a la entrada 1$\overline{PRE}$ o a la 1$\overline{CLR}$, el flip-flop pasa a estado SET o RESET independientemente del estado del resto de las entradas.

Problema relacionado. Determinar la forma de onda de salida 1Q si se intercambian las señales 1$\overline{PRE}$ y 1$\overline{CLR}$.

REVISIÓN DE LA SECCIÓN 8.2

1. Describir la principal diferencia entre un latch S-R con entrada de habilitación y un flip-flop S-R disparado por flanco.
2. ¿Cuál es la diferencia en el funcionamiento básico entre un flip-flop J-K y un flip-flop S-R?
3. Suponer que el flip-flop de la Figura 8.21 es del tipo disparado por flanco negativo. Describir la forma de onda de salida, para las mismas señales de reloj y datos (CLK y D).

8.3 ■ FLIP-FLOPS MAESTRO-ESCLAVO

Otra clase de flip-flops son los maestro-esclavo disparados por pulso, que han sido reemplazados progresivamente por los dispositivos disparados por flanco. Aunque los flip-flops maestro-esclavo están prácticamente obsoletos, todavía es posible encontrar este tipo de flip-flop en algunos equipos. Los datos se introducen en el flip-flop con el flanco anterior del impulso del reloj, pero la salida no refleja el estado de la entrada hasta que llega el flanco posterior. El flip-flop maestro-esclavo disparado por pulso no permite variar los datos mientras el impulso del reloj se encuentre activo.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Identificar un flip-flop maestro-esclavo mediante su símbolo lógico. ■ Explicar en qué se diferencia un flip-flop maestro-esclavo de los dispositivos disparados por flanco. ■ Comentar el funcionamiento básico de los flip-flops maestro-esclavo disparados por pulso.

El flip-flop J-K maestro-esclavo disparado por pulso

En la Figura 8.32 se muestra un flip-flop **maestro-esclavo** J-K. La tabla de verdad es la misma que la de los flip-flops J-K disparados por flanco, excepto en la manera en que se sincroniza con la señal de reloj. Internamente es, sin embargo, bastante diferente.

Este tipo de flip-flop se compone de dos secciones: el maestro y el esclavo. El maestro es básicamente un latch con entrada de habilitación y el esclavo es lo mismo, excepto en que está sincronizado con el impulso invertido del reloj y se controla mediante las salidas del maestro en lugar de por las entradas externas J-K.

El maestro va a asumir el estado determinado por las entradas J y K durante el flanco anterior (positivo) del impulso de reloj. El estado del maestro se transfie-

re entonces al esclavo durante el flanco posterior (negativo) del impulso de reloj, ya que las salidas del maestro se aplican a las entradas del esclavo y se invierte el impulso de reloj en el esclavo. En el flanco de bajada del impulso de reloj, el estado del esclavo aparece en las salidas Q y $\overline{Q}$. La salida Q se realimenta a la entrada de la puerta G_2 y la salida $\overline{Q}$ se realimenta a la entrada de la puerta G_1 para producir el modo de basculación característico cuando $J = 1$ y $K = 1$. El funcionamiento lógico se resume en la Tabla 8.5. Una limitación del funcionamiento maestro-esclavo es que las entradas (J y K) no pueden cambiar mientras el impulso de reloj está activo, porque el estado del latch maestro puede cambiar durante dicho intervalo de tiempo.

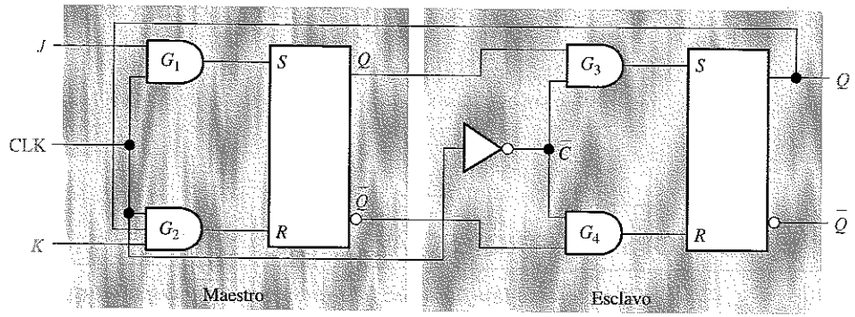


FIGURA 8.32
Diagrama lógico de un flip-flop J-K maestro-esclavo.

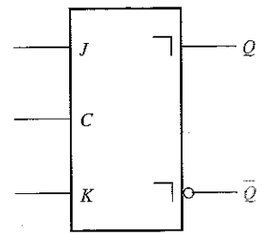
TABLA 8.5
Tabla de verdad de un flip-flop J-K maestro-esclavo.

Entradas			Salidas		
J	K	CLK	Q	$\overline{Q}$	Observaciones
0	0	⊓	Q_0	$\overline{Q}_0$	No cambio
0	1	⊓	0	1	RESET
1	0	⊓	1	0	SET
1	1	⊓	$\overline{Q}_0$	Q_0	Basculación

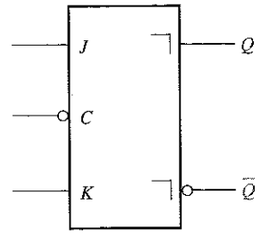
⊓ = pulso de reloj

Q_0 = nivel de salida anterior al pulso de reloj

Los símbolos lógicos del flip-flop J-K maestro-esclavo se muestran en la Figura 8.33. La clave para identificar un flip-flop disparado por pulso (maestro-esclavo) mediante su símbolo lógico es el símbolo de *salida pospuesta* ANSI/IEEE (⌉) situado en las salidas. Este símbolo significa que la salida no refleja los datos de la entrada *J-K*, hasta que llega el flanco de reloj (bien positivo o negativo) siguiente al flanco disparador. Téngase en cuenta que no hay ningún indicador dinámico de entrada (▷) en la entrada del reloj (C), como ocurre en un flip-flop disparado por flanco.



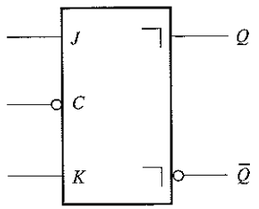
(a) Reloj activo a nivel ALTO: los datos se sincronizan con el flanco positivo del impulso de reloj y se transfieren a la salida en el siguiente flanco negativo



(b) Reloj activo a nivel BAJO: los datos se sincronizan con el flanco negativo del impulso de reloj y se transfieren a la salida en el siguiente flanco positivo

FIGURA 8.33
Símbolos lógicos del flip-flop J-K (maestro-esclavo) disparado por pulso.

EJEMPLO 8.10 Determinar la salida Q del flip-flop J-K maestro-esclavo para las formas de onda de entrada mostradas en la Figura 8.34(a). El flip-flop se inicia en estado RESET y el reloj es activo a nivel BAJO.



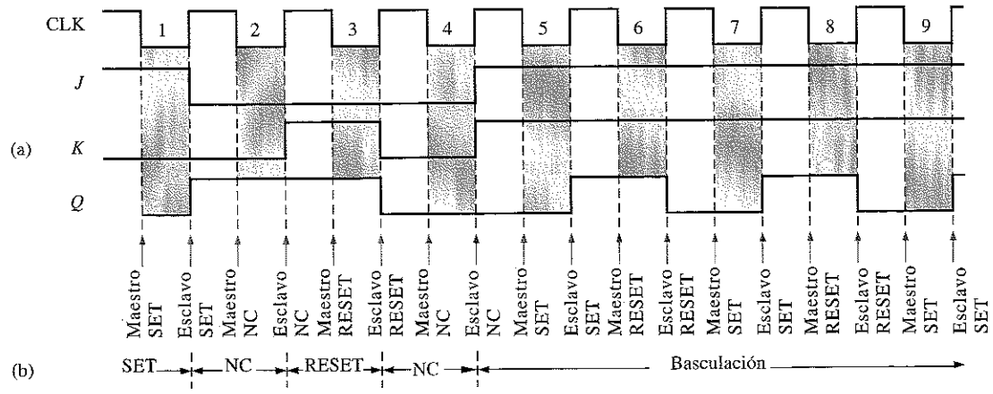


FIGURA 8.34

Solución. La forma de onda Q se puede ver en la Figura 8.34(b). Los estados de entrada y la situación en que se encuentra el circuito al principio y final de cada impulso de reloj se etiquetan para demostrar su funcionamiento. NC indica el modo de no cambio.

Problema relacionado. ¿Cómo sería la salida Q si se invierten las formas de onda J y K?

REVISIÓN DE LA SECCIÓN 8.3

1. Describir la diferencia principal entre los flip-flops disparados por pulso y los disparados por flanco.
2. Suponer que la entrada D de un flip-flop cambia de nivel BAJO a nivel ALTO en mitad de un pulso positivo de reloj.
   (a) Describir qué ocurrirá si se trata de un flip-flop disparado por flanco.
   (b) Describir qué ocurrirá si se trata de un flip-flop maestro-esclavo disparado por pulso.

8.4 ■ CARACTERÍSTICAS DE OPERACIÓN DE LOS FLIP-FLOPS

El funcionamiento, requisitos de operación y limitaciones de los flip-flops se especifican mediante varias características de funcionamiento o parámetros que se encuentran en las hojas de características del dispositivo. Generalmente, las especificaciones son aplicables a todos los flip-flops CMOS y TTL.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Definir retardo de propagación. ■ Explicar las características de distintos retardos de propagación. ■ Definir *tiempo de establecimiento* y explicar en qué limita el funcionamiento de los flip-flops. ■ Definir *tiempo de mantenimiento* y explicar en qué limita el funcionamiento de los flip-flops. ■ Explicar el significado de la frecuencia máxima de reloj. ■ Explicar las distintas especificaciones de los anchos de los impulsos. ■ Definir *disipación de potencia* y calcular su valor en un dispositivo determinado. ■ Comparar varias series de flip-flops en función de sus parámetros de funcionamiento.

Retardos de propagación

Se define **retardo de propagación** como el intervalo de tiempo requerido para que se produzca un cambio en la salida una vez que se ha aplicado una señal en la entrada. Existen distintas categorías de retardos de propagación que son importantes en el funcionamiento de los flip-flops:

1. El retardo de propagación t_{PLH} se mide desde el flanco de disparo del impulso de reloj hasta la transición de nivel BAJO a nivel ALTO de la salida. Este retardo se ilustra en la Figura 8.35(a).
2. El retardo de propagación t_{PHL} se mide desde el flanco de disparo de impulso del reloj hasta la transición de nivel ALTO a nivel BAJO de la salida. Este retardo se ilustra en la Figura 8.35(b).

3. El retardo de propagación t_{PLH} medido desde la entrada de inicialización (preset) hasta la transición de nivel BAJO a nivel ALTO de la salida. Este retardo se ilustra en la Figura 8.36(a), para una entrada de inicialización activa a nivel BAJO.

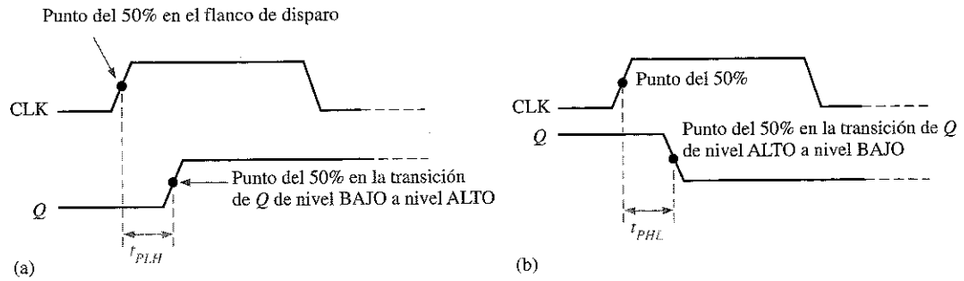


FIGURA 8.35
Retardos de propagación entre el reloj y la salida.

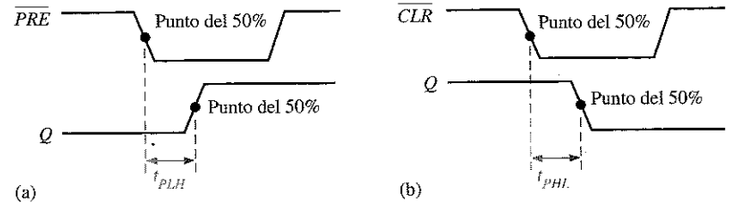


FIGURA 8.36
Retardos de propagación entre la entrada de inicialización y la salida y entre la entrada de borrado y la salida.

4. El retardo de propagación t_{PHL} medido desde la entrada de borrado (clear) hasta la transición de nivel ALTO a nivel BAJO de la salida. Este retardo se ilustra en la Figura 8.35(b), para una entrada de borrado activa a nivel BAJO.

Tiempo de set-up (establecimiento)

El **tiempo de establecimiento,** *setup time,* (t_S) es el intervalo mínimo que los niveles lógicos deben mantener constantes en las entradas (J y K, S y R o D) antes de que llegue el flanco de disparo del impulso de reloj, de modo que dichos niveles sincronicen correctamente en el flip-flop. Este intervalo, para el caso de un flip-flop D, se muestra en la Figura 8.37.

Tiempo de hold (mantenimiento)

El **tiempo de mantenimiento,** *hold time,* (t_h) es el intervalo mínimo que los niveles lógicos deben mantenerse constantes en las entradas después de que haya pasado el

flanco de disparo del impulso de reloj, de modo que dichos niveles se sincronicen correctamente en el flip-flop. Esto se ilustra, para el caso de un flip-flop D, en la Figura 8.38.

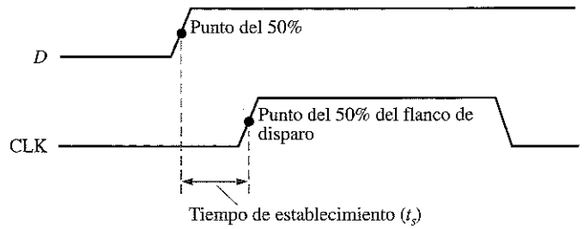


FIGURA 8.37
Tiempo de establecimiento (t_s).

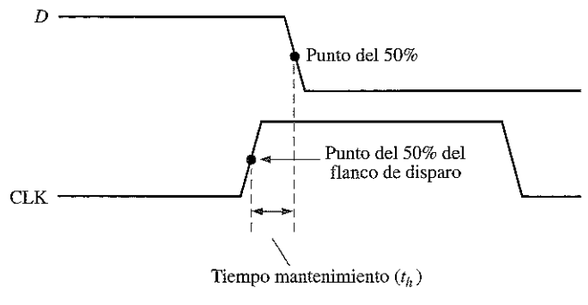


FIGURA 8.38
Tiempo de mantenimiento (t_h).

Frecuencia máxima de reloj

La frecuencia máxima de reloj ($f_{máx}$) es la mayor velocidad a la que se puede disparar el flip-flop de manera fiable. Para frecuencias del reloj por encima de la máxima, el flip-flop puede ser incapaz de responder lo suficientemente rápido y su funcionamiento se vería deteriorado.

Anchura de los impulsos

Usualmente, los fabricantes especifican la anchura mínima de los impulsos (t_W) para un funcionamiento adecuado de las entradas de reloj, inicialización y borrado. Típicamente, el reloj se especifica mediante sus intervalos de tiempo mínimo para los niveles ALTO y BAJO.

Disipación de potencia

La **disipación de potencia** de cualquier circuito digital se define como la potencia total consumida por el dispositivo. Por ejemplo, si el flip-flop funciona con una fuente de continua de +5 V y circula por él una corriente de 5 mA, la disipación de potencia es:

$$P = V_{CC} \times I_{CC} = 5\ \text{V} \times 5\ \text{mA} = 25\ \text{mW}$$

Esta disipación de potencia es muy importante en la mayoría de las aplicaciones en las que la capacidad de la fuente de continua (DC) juegue un papel importante. Como ejemplo, vamos a suponer que tenemos un sistema digital que requiere un total de diez flip-flops, y que cada uno de ellos disipa una potencia de 25 mW. El requisito de disipación de potencia total es:

$$P_T = 10 \times 25\ \text{mW} = 250\ \text{mW} = 0{,}25\ \text{W}$$

Esto nos dice cuál es la potencia de salida necesaria de nuestra fuente de alimentación. Si los flip-flops funcionan con +5 V de continua, entonces la corriente total que tiene que suministrar la fuente es la siguiente:

$$I = \frac{250\,\text{mW}}{5\,\text{V}} = 50\ \text{mA}$$

Tenemos que utilizar una fuente de +5 V que sea capaz de proporcionar al menos 50 mA de corriente.

Comparación de flip-flops específicos

La Tabla 8.6 proporciona una comparación para cuatro flip-flops TTL y CMOS del mismo tipo, en función de los distintos parámetros de funcionamiento discutidos en esta sección.

TABLA 8.6
Comparación de los parámetros de funcionamiento para cuatro familias de CI flip-flop del mismo tipo a 25 °C.

Parámetros (tiempos en ns)	CMOS		TTL	
	74HC74A	74AHC74	74LS74A	74F74
t_{PHL} (CLK a Q)	17	4,6	40	6,8
t_{PLH} (CLK a Q)	17	4,6	25	8,0
t_{PHL} ($\overline{CLR}$ a Q)	18	4,8	40	9,0
t_{PLH} ($\overline{PRE}$ a Q)	18	4,8	25	6,1
t_S (tiempo de setup)	14	5,0	20	2,0
t_h (tiempo de hold)	3,0	0,5	5	1,0
t_W (CLK HIGH)	10	5,0	25	4,0
t_W (CLK LOW)	10	5,0	25	5,0
t_W ($\overline{CLR}$ /$\overline{PRE}$)	10	5,0	25	4,0
$f_{máx}$ (MHz)	35	170	25	100
Potencia (mW)	0,012	1,1		
Potencia (mW), ciclo de trabajo 50%			44	88

REVISIÓN DE LA SECCIÓN 8.4

1. Definir los siguientes parámetros:
   (a) tiempo de establecimiento **(b)** tiempo de mantenimiento
2. ¿Cuál de todos los flip-flops de la Tabla 8.6 puede funcionar a mayor frecuencia?

Consejos Prácticos

Una ventaja de los dispositivos CMOS es que pueden operar en un rango más amplio de tensiones continuas de alimentación (normalmente de 2 V a 6 V) que los dispositivos TTL, y, por tanto, se pueden emplear fuentes de alimentación más baratas que no necesitan una regulación precisa. Para los circuitos CMOS también se puede utilizar baterías como fuentes de alimentación primarias o secundarias. Además, tensiones más bajas significan que el CI disipa menos potencia. El inconveniente es que el rendimiento de un circuito CMOS se degrada con tensiones de alimentación bajas. Por ejemplo, la frecuencia máxima de reloj garantizada de un flip-flop CMOS es mucho menor para $V_{CC} = 2$ V que para $V_{CC} = 6$ V.

8.5 ■ APLICACIONES DE LOS FLIP-FLOPS

En esta sección, se describen tres aplicaciones de carácter general de los flip-flops que nos van a proporcionar una idea básica de cómo pueden utilizarse. En los Capítulos 9 y 10 se tratarán en más detalle las aplicaciones de los flip-flops en contadores y registros.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar la aplicación de los flip-flops en el almacenamiento de datos. ■ Describir cómo se emplean los flip-flops para la división de frecuencia. ■ Explicar cómo se usan los flip-flops en aplicaciones básicas de contadores.

Ejemplos de aplicación

Almacenamiento de datos paralelo. Uno de los requisitos más comunes de los sistemas digitales consiste en almacenar de forma simultánea una serie de bits de datos, procedentes de varias líneas paralelas, en un grupo de flip-flops. Este proceso se ilustra en la Figura 8.39(a), utilizando cuatro flip-flops. Cada una de las cuatro líneas paralelas de datos se conecta a la entrada D de un flip-flop. Las entradas de reloj de los flip-flops se conectan juntas, de forma que los flip-flops son disparados mediante el mismo impulso del reloj. En este ejemplo, se utilizan flip-flops disparados por flanco positivo, por lo que los datos de las entradas D se almacenan simultáneamente en los flip-flops con el flanco positivo de reloj, como se indica en el diagrama de tiempos de la Figura 8.39(b). Además, las entradas de puesta a cero asíncronas (R) se conectan a una línea $\overline{CLR}$ común, que inicialmente pone a cero a todos los flip-flops.

Este grupo de cuatro flip-flops es un ejemplo de un sencillo registro utilizado para almacenamiento de datos. En los sistemas digitales, los datos se almacenan normalmente en grupos de bits (usualmente ocho o múltiplos de ocho) que representan números, códigos u otras informaciones. Los registros se tratarán en detalle en el Capítulo 10.

División de frecuencia. Otra de las aplicaciones de un flip-flop es la división (reducción) de frecuencia de una señal periódica. Cuando se aplica un tren de impulsos a la entrada de reloj de un flip-flop J-K conectado en modo de basculación ($J = K = 1$), la salida Q es una señal cuadrada que tiene una frecuencia igual a la mitad de la

que tiene la señal de reloj. Por consiguiente, se puede utilizar un único flip-flop como un divisor por 2, como muestra la Figura 8.40. Como puede verse, el flip-flop cambia de estado en cada flanco de disparo del impulso de reloj (flancos positivos en este caso). Esto da lugar a una salida que varía a la frecuencia mitad de la señal de reloj.

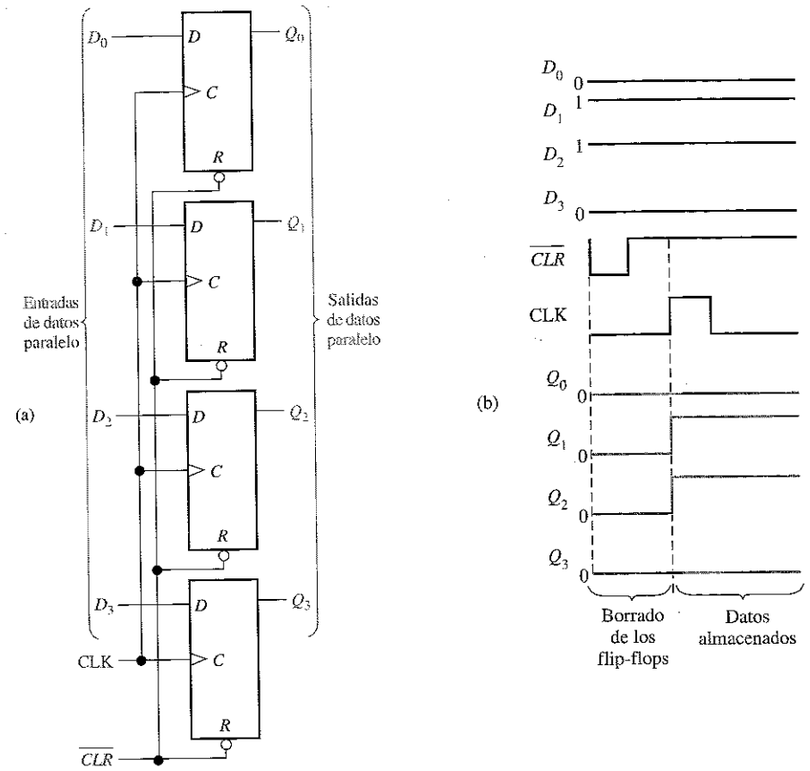


FIGURA 8.39
Ejemplo de utilización de flip-flops en un registro sencillo para almacenamiento paralelo de datos.

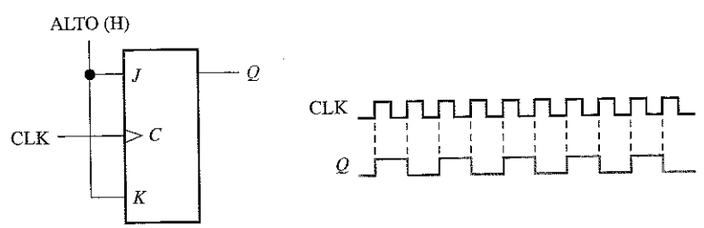


FIGURA 8.40
El flip-flop J-K como dispositivo divisor por 2.

Se pueden conseguir divisiones sucesivas de la frecuencia del reloj conectando la salida de un flip-flop a la entrada de reloj de un segundo flip-flop, como se muestra en la Figura 8.41. El flip-flop B divide la frecuencia de la salida Q_A por 2. La salida Q_B es, por lo tanto, un cuarto de la frecuencia de la señal de reloj original. En estos cronogramas no se muestran los retardos de propagación.

Si se conectan varios flip-flops de esta manera, se puede conseguir una división de frecuencias de 2^n, donde n es el número de flip-flops. Por ejemplo, tres flip-flops dividen la frecuencia de reloj por $2^3 = 8$; cuatro flip-flops dividen la frecuencia de reloj por $2^4 = 16$, y así sucesivamente.

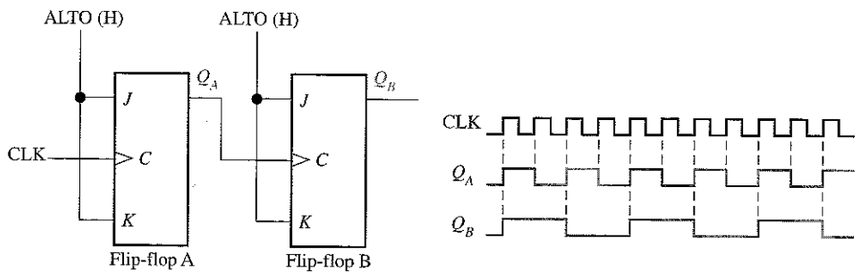


FIGURA 8.41
Ejemplo de utilización de dos flip-flops J-K para dividir la frecuencia de reloj por 4.

EJEMPLO 8.11 Desarrollar la forma de onda f_{out} para el circuito de la Figura 8.42, cuando se aplica una señal cuadrada de 8 kHz en la entrada de reloj del flip-flop A.

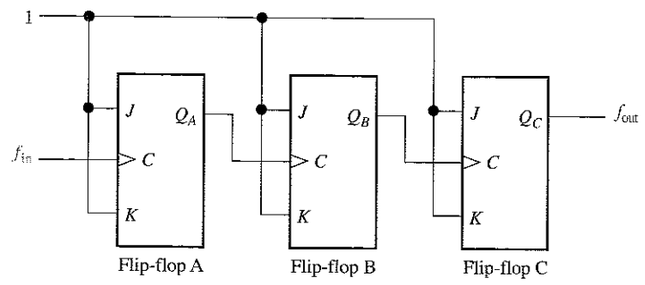


FIGURA 8.42

Solución. Los tres flip-flops están conectados para dividir la frecuencia de entrada por ocho ($2^3 = 8$) y la señal f_{out} se muestra en la Figura 8.43. Dado que se trata de flip-flops disparados por flanco positivo, las salidas cambian durante el flanco positivo del reloj. Hay un impulso de salida por cada ocho impulsos de entrada, de forma que la frecuencia de salida es 1 kHz. Las señales Q_A y Q_B también se muestran.

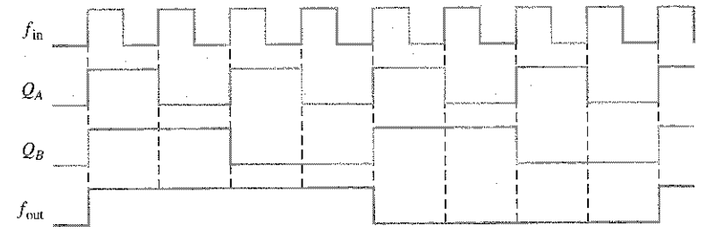


FIGURA 8.43

Problema relacionado. ¿Cuántos flip-flops se requieren para dividir una frecuencia entre treinta y dos?

Contadores. Otra de las aplicaciones importantes de los flip-flops son los contadores digitales, que serán tratados en detalle en el próximo capítulo. El concepto se ilustra en la Figura 8.44. Los flip-flops son de tipo J-K disparados por flanco negativo. Ambos flip-flops se encuentran inicialmente en estado RESET. El flip-flop A bascula en las transiciones negativas de cada impulso de reloj. La salida Q del flip-flop A dispara el flip-flop B, de manera que siempre que Q_A realiza una transición de nivel ALTO a nivel BAJO, el flip-flop B bascula. Las señales resultantes Q_A y Q_B se muestran en la figura.

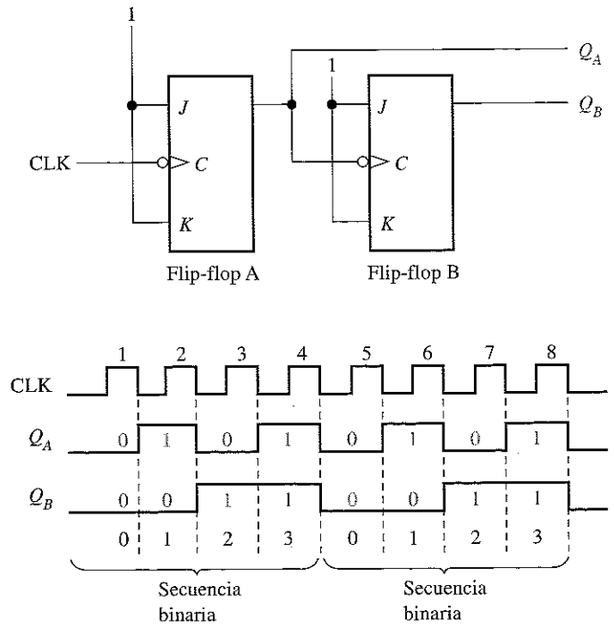


FIGURA 8.44
Flip-flops utilizados para generar una secuencia de cuenta binaria. Se muestran dos repeticiones (00, 01, 10, 11).

Obsérvese la secuencia de Q_A y Q_B en la Figura 8.44. Previamente al impulso de reloj 1, $Q_A = 0$ y $Q_B = 0$; tras el impulso de reloj 1, $Q_A = 1$ y $Q_B = 0$; después del impulso de reloj 2, $Q_A = 0$ y $Q_B = 1$, y tras el impulso de reloj 3, $Q_A = 1$ y $Q_B = 1$. Si se toma Q_A como el bit menos significativo, se produce una secuencia binaria de dos bits a medida que se disparan los flip-flops. Esta secuencia binaria se repite cada cuatro impulsos de reloj, como se muestra en el diagrama de tiempos de la Figura 8.44. Por tanto, los flip-flops siguen una secuencia de 0 a 3 (00, 01, 10, 11) y luego vuelven a 0 para comenzar la misma secuencia de nuevo.

EJEMPLO 8.12 Determinar la forma de onda de salida en función del reloj para Q_A, Q_B y Q_C en el circuito de la Figura 8.45 y mostrar la secuencia binaria representada por estas señales.

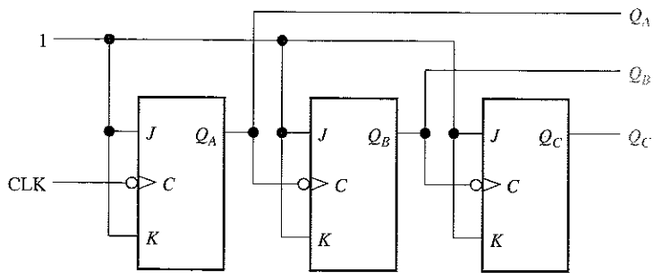


FIGURA 8.45

Solución. El diagrama de tiempos que se obtiene se muestra en la Figura 8.46. Obsérvese que las salidas cambian en los flancos negativos de los impulsos de reloj. Las salidas siguen la secuencia binaria 000, 001, 010, 011, 100,101, 110 y 111, tal y como se indica.

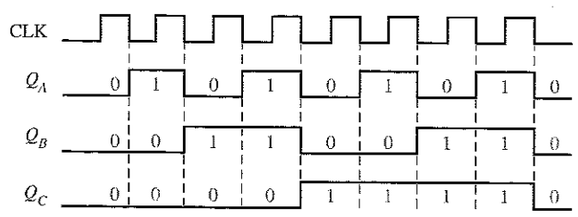


FIGURA 8.46

Problema relacionado. ¿Cuántos flip-flops son necesarios para producir una secuencia binaria que represente los números decimales de 0 a 15?

REVISIÓN DE LA SECCIÓN 8.5

1. Un conjunto de flip-flops utilizado para almacenamiento de datos se denomina _____.
2. ¿Cómo se tiene que conectar un flip-flop J-K para funcionar como un dispositivo divisor por 2?
3. ¿Cuántos flip-flops son necesarios para obtener un dispositivo divisor por 64?

8.6 ■ MONOESTABLES

Los monoestables son dispositivos multivibradores que sólo tienen un único estado estable. Normalmente, un monoestable se encuentra en su estado estable, cambiando a su estado inestable sólo cuando se dispara. Una vez que se ha disparado, el monoestable permanece en su estado inestable durante un determinado intervalo de tiempo, volviendo a continuación a su estado estable. El tiempo que este dispositivo permanece en el estado inestable determina la anchura del impulso de su salida.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir el funcionamiento básico de un monoestable. ■ Explicar cómo funciona un monoestable no redisparable. ■ Explicar cómo funciona un monoestable redisparable. ■ Configurar los monoestables 74121 y 74LS122 para obtener una anchura de impulso determinada. ■ Reconocer el símbolo de un trigger Schmitt y explicar qué significa.

La Figura 8.47 muestra un **monoestable** (multivibrador de un solo estado) básico formado por una puerta lógica y un inversor. Cuando se aplica un impulso a la entrada **de disparo** (trigger), la salida de la puerta G_1 pasa a nivel BAJO. Esta transición de nivel ALTO a nivel BAJO se acopla por medio del condensador a la entrada del inversor G_2. La presencia de un aparente nivel BAJO en G_2 hace que su salida pase a nivel ALTO. Este nivel ALTO se realimenta a la puerta G_1, manteniendo su salida a nivel BAJO. Hasta este punto, el impulso de disparo ha hecho que la salida del monoestable, Q, sea un nivel ALTO.

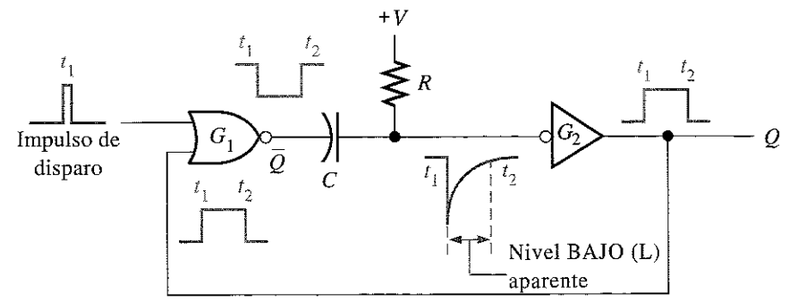


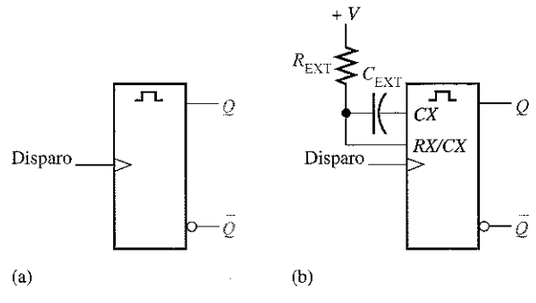
FIGURA 8.47
Un sencillo circuito monoestable.

El condensador comienza inmediatamente a cargarse a través de R hasta alcanzar su tensión máxima. La velocidad de carga está determinada por la constante de tiempo RC. Cuando el condensador se carga hasta un determinado nivel, el cual aparece como un nivel ALTO en G_2, la salida pasa de nuevo a nivel BAJO.

En resumen, la salida del inversor G_2 pasa a nivel ALTO en respuesta a la entrada de disparo. Permanece a nivel ALTO durante un tiempo definido por la constante de tiempo, RC, y al final de este intervalo pasa a nivel BAJO. De esta manera, *un único impulso estrecho produce un único impulso de salida cuyo periodo se controla mediante la constante de tiempo RC*. Este modo de operación se puede ver en la Figura 8.47.

En la Figura 8.48(a) se muestra el típico símbolo lógico de un monoestable, y en la 8.48(b) se presenta el mismo símbolo con R y C externos. Los dos tipos fundamentales de circuitos integrados monoestables son los redisparables y los no redisparables.

FIGURA 8.48
Símbolos lógicos básicos de los monoestables. CX y RX indican componentes externos.

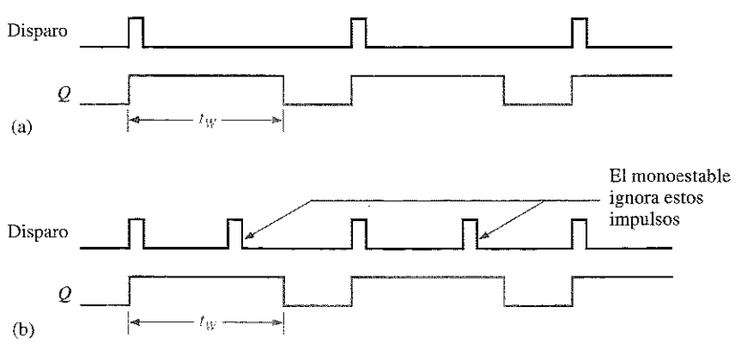


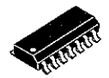
Monoestables no redisparables

Un monoestable no redisparable no responderá a ningún impulso de disparo adicional, desde el momento en que se pasa a su estado inestable hasta que retorna a su estado estable. En otras palabras, ignorará cualquier impulso de disparo que ocurra antes de que termine el periodo inestable. El tiempo que permanece el monoestable en su estado inestable es la anchura del impulso de salida.

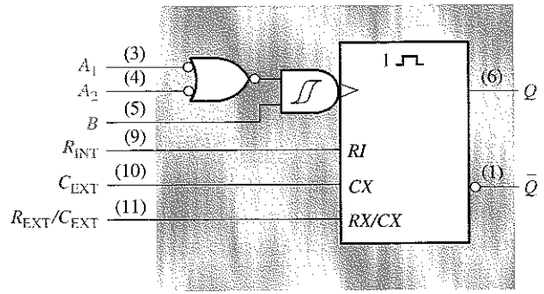
La Figura 8.49 presenta un monoestable no redisparable, disparado a intervalos mayores y menores que su anchura de impulso. Obsérvese que, en el segundo caso, los impulsos adicionales se ignoran.

FIGURA 8.49
Funcionamiento de un monoestable no redisparable.

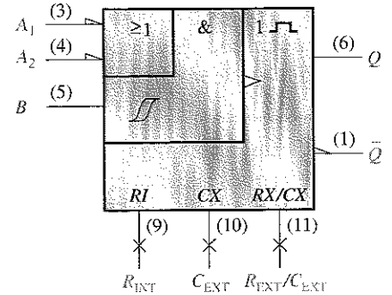




Un monoestable no redisparable. El 74121 es un ejemplo de un circuito monoestable integrado no redisparable. Como muestra la Figura 8.50, está previsto para conectarse a R y C externos. Las entradas etiquetadas como A_1, A_2 y B son entradas de activación de disparo. La entrada R_{INT} está conectada a una resistencia interna de temporización de 2 kΩ.



(a) Símbolo lógico tradicional



(b) Símbolo lógico del estándar ANSI/IEEE 91-1984 (×= conexión no lógica). "1 ⎍" es el símbolo de cualificación de un monoestable no redisparable

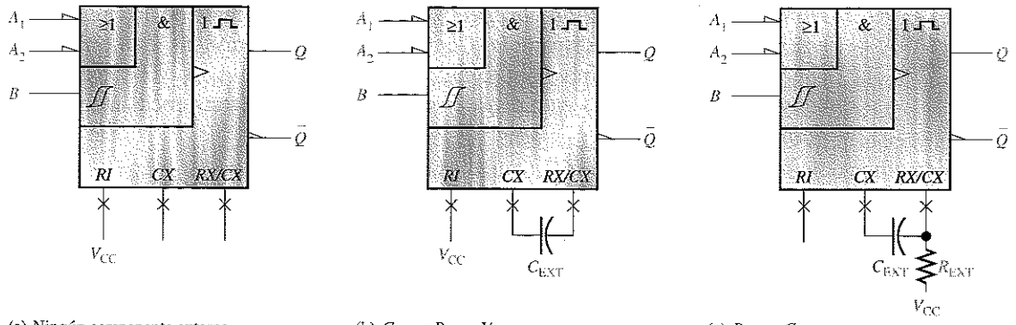
FIGURA 8.50
Símbolos lógicos del monoestable no redisparable 74121.

Establecimiento de la anchura del impulso. Cuando no se utiliza ningún componente de temporización externo y la resistencia de temporización interna (R_{INT}) se conecta a V_{CC}, como se muestra en la Figura 8.51(a) se produce un impulso típico de unos 30 ns de anchura. La anchura del impulso se puede ajustar entre 30 ns y 28 s utilizando los componentes externos. La Figura 8.51(b) muestra la conexión de una resistencia interna (2 kΩ) y un condensador externo. La parte (c) ilustra la conexión de una resistencia y un condensador externos. La anchura del impulso de salida se ajusta mediante los valores de la resistencia (R_{INT} = 2 kΩ, R_{EXT} variable) y del condensador de acuerdo con la siguiente fórmula:

$$t_W = 0.7\, R\, C_{EXT} \qquad \textbf{(8.1)}$$

donde R puede ser tanto R_{INT} como R_{EXT}. Cuando R se expresa en kilohmios (kΩ) y C_{EXT} en picofaradios (pF), la anchura del impulso de salida t_W se obtiene en nanosegundos (ns).

El símbolo del trigger Schmitt. El símbolo ⎎ indica una entrada de un trigger Schmitt. Este tipo de entrada utiliza un circuito umbral específico que produce **histéresis**, una característica que previene la conmutación errática entre estados cuando una tensión de disparo que varía muy lentamente se encuentra en las cercanías de un nivel de entrada crítico. Esto permite que se produzcan disparos fiables incluso cuando la entrada esté variando tan lentamente como a 1 voltio/segundo.



(a) Ningún componente externo
R_{INT} a V_{CC}
$t_W \cong 30$ ns

(b) C_{EXT} y R_{INT} a V_{CC}
$t_W = 0{,}7(2\ k\Omega)C_{EXT}$

(c) R_{EXT} y C_{EXT}
$t_W = 0{,}7R_{EXT}\ C_{EXT}$

FIGURA 8.51
Tres maneras de ajustar la anchura de los impulsos en un 74121.

EJEMPLO 8.13 Una determinada aplicación requiere un monoestable con una anchura de impulso de aproximadamente 100 ms. Utilizando un 74121, dibujar las conexiones y hallar los valores de sus componentes.

Solución. Arbitrariamente se selecciona $R_{EXT} = \mathbf{39\ k\Omega}$ y se calcula la capacidad necesaria.

$$t_W = 0{,}7R_{EXT}C_{EXT}$$

$$C_{EXT} = \frac{t_W}{0{,}7R_{EXT}}$$

donde C_{EXT} está en pF, R_{EXT} en kΩ y t_W en ns. Ya que 100 ms = 1×10^8 ns,

$$C_{EXT} = \frac{1 \times 10^8\ \text{ns}}{0{,}7\ (39\ \text{k}\Omega)} = 3{,}66 \times 106\ \text{pF} = \mathbf{3{,}66\ \mu F}$$

Un condensador estándar de 3,3 μF proporcionará una anchura de impulso de salida de 91 ms. Las conexiones adecuadas se muestran en la Figura 8.52. Para conseguir una anchura de impulso más próxima a 100 ms, se pueden probar otras combinaciones de valores para R_{EXT} y C_{EXT}. Por ejemplo, R_{EXT} = 68 kΩ y C_{EXT} = 2,2 μF proporciona una anchura de impulso de 105 ms.

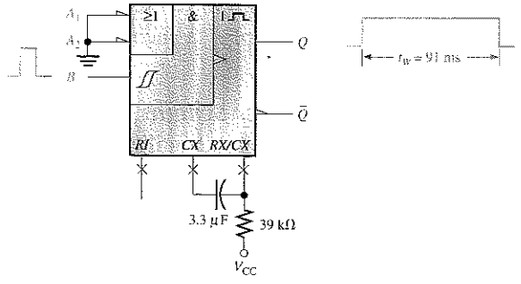


FIGURA 8.52

Problema relacionado. Utilizar un condensador externo junto con R_{INT}, para producir un impulso de salida de anchura 10 μs en un 74121.

Monoestables redisparables

Un monoestable redisparable puede ser disparado antes de que retorne a su estado estable. El resultado del redisparo es una ampliación de la anchura del impulso, como se muestra en la Figura 8.53.

Monoestable redisparable. El 74LS122 es un ejemplo de un CI monoestable redisparable con entrada de borrado (clear). También está diseñado para añadir R y C externas, como muestra la Figura 8.54. Las entradas etiquetadas como A_1, A_2, B_1 y B_2 son entradas de activación de disparo.

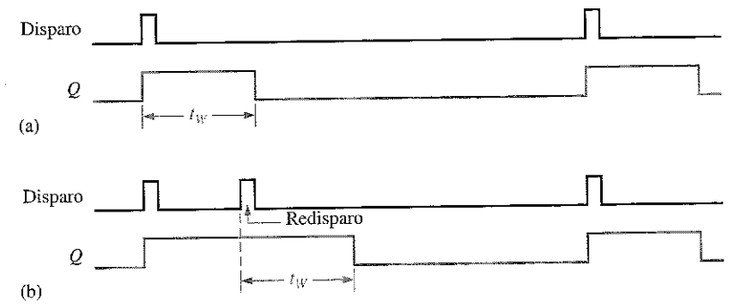
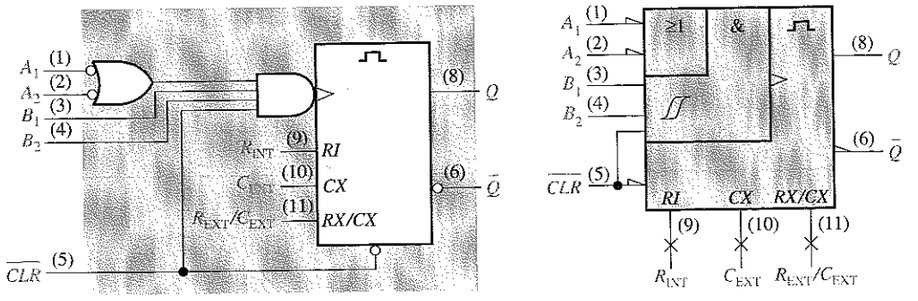


FIGURA 8.53
Funcionamiento de un monoestable redisparable.



(a) Símbolo lógico tradicional

(b) Símbolo lógico del ANSI/IEEE std. 91-1984 (× = conexión no lógica). ⊓ es el símbolo de cualificación de un monoestable redisparable

FIGURA 8.54
Símbolo lógico del monoestable no redisparable 74LS122.

Sin ningún componente adicional se obtiene un impulso de unos 45 ns de anchura. Se pueden conseguir impulsos más anchos mediante el uso de componentes externos. La fórmula general para calcular los valores de estos componentes para una determinada anchura del impulso (t_W) es:

$$t_W = 0{,}32RC_{EXT}\left(1 + \frac{0{,}7}{R}\right) \qquad \textbf{(8.2)}$$

donde 0,32 es una constante determinada por el tipo particular de monoestable, R se expresa en kΩ y puede ser tanto la resistencia interna como la externa, C_{EXT} se expresa en pF y t_W en ns. La resistencia interna vale 10 kΩ y puede utilizarse en lugar de una resistencia externa; obsérvese la diferencia entre esta fórmula y la correspondiente del 74121, mostrada en la Ecuación (8.1).

EJEMPLO 8.14 Determinar los valores de R_{EXT} y C_{EXT} que producen un impulso de anchura 1μs, cuando se conectan a un 74LS122.

Solución. Suponer un valor de C_{EXT} = **560 pF** y luego calcular R_{EXT}. La anchura del impulso t_W debe expresarse en ns y C en pF. R_{EXT} se obtendrá en kΩ.

$$t_W = 0{,}32R_{EXT}C_{EXT}\left(1 + \frac{0{,}7}{R_{EXT}}\right) = 0{,}32R_{EXT}C_{EXT} + 0{,}7\left(\frac{0{,}32_{EXT}C_{EXT}}{R_{EXT}}\right)$$

$$= 0{,}32R_{EXT}C_{EXT} + (0{,}7)(0{,}32)C_{EXT}$$

$$R_{EXT} = \frac{t_W - (0{,}7)(0{,}32)C_{EXT}}{0{,}32C_{EXT}} = \frac{t_W}{0{,}32C_{EXT}} - 0{,}7$$

$$= \frac{1000\text{ ns}}{(0{,}32)560\text{ pF}} - 0{,}7 = \mathbf{4{,}88\ k\Omega}$$

Utilizar un valor estándar de **4,7 kΩ**

Problema relacionado. Dibujar las conexiones y calcular los valores de los distintos componentes para un monoestable 74LS122, que tiene un impulso de salida de 5 μs de anchura. Suponer que C_{EXT} = 560 pF.

Ejemplo de aplicación

Una aplicación práctica de los monoestables es un temporizador secuencial, que puede utilizarse para encender una serie de luces. Este tipo de circuito puede ser empleado, por ejemplo, en una autopista en construcción para indicar cambios sucesivos de carril.

La Figura 8.55 nos muestra tres monoestables 74LS122 conectados como temporizador secuencial. Este circuito particular produce una secuencia de tres impulsos

de 1 segundo. El primer monoestable se dispara mediante el cierre de un interruptor o mediante un impulso de entrada de baja frecuencia, que produce un impulso de salida de 1 segundo. Cuando el primer monoestable (ME 1) pasa a su estado estable y pasa a nivel BAJO, el segundo monoestable (ME 2) se dispara, produciendo también un impulso de salida de 1 segundo. Cuando este segundo impulso pasa a nivel BAJO, el tercer monoestable (ME 3) se dispara, produciéndose el tercer impulso de 1 segundo. El diagrama de tiempos de salida se muestra en la figura. Se pueden realizar variaciones de esta configuración básica para producir diversas salidas de temporización.

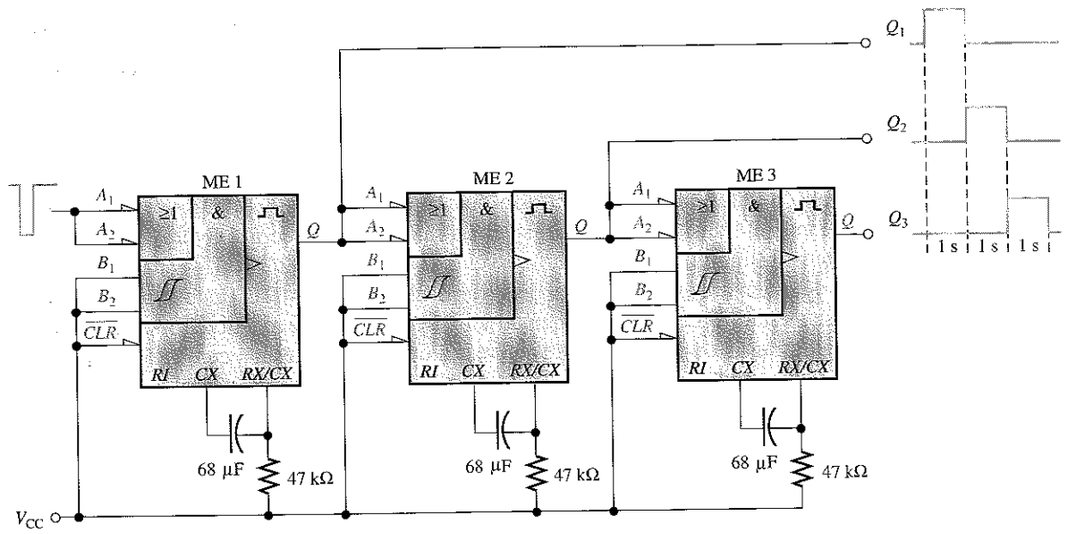


FIGURA 8.55
Circuito de temporización secuencial utilizando tres monoestables 74LS122.

REVISIÓN DE LA SECCIÓN 8.6

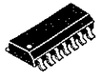
1. Describir las diferencias entre un monoestable no redisparable y uno redisparable.
2. ¿Cómo se ajusta la anchura del impulso en la mayoría de los circuitos integrados monoestables?

8.7 ■ EL TEMPORIZADOR 555

El temporizador 555 es un dispositivo versátil y muy utilizado, porque puede ser configurado de dos modos distintos, bien como multivibrador monoestable o como multivibrador aestable (oscilador). Un multivibrador aestable no tiene estados estables y varía, por tanto, una y otra vez (oscila) entre dos estados inestables, sin utilizar un circuito de disparo externo.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir los elementos básicos de un temporizador 555. ■ Configurar un temporizador 555 como monoestable. ■ Configurar un temporizador 555 como oscilador.



Funcionamiento básico

En la Figura 8.56 se muestra un diagrama funcional con los componentes internos de un **temporizador** 555. Los comparadores son dispositivos cuyas salidas están a nivel ALTO cuando la tensión en la entrada positiva (+) es mayor que la tensión en la entrada negativa (-), y están a nivel BAJO cuando la tensión de entrada negativa es mayor que la tensión de entrada positiva. El divisor de tensión, formado por tres resistencias de 5kΩ, proporciona un nivel de disparo de $1/3V_{CC}$ y un nivel umbral de $2/3V_{CC}$. La entrada de la tensión de control (pin 5) se puede emplear para ajustar externamente los niveles de disparo y umbral a otros valores en caso necesario. Cuando la entrada de disparo, normalmente a nivel ALTO, desciende momentáneamente por debajo de $1/3\ V_{CC}$, la salida del comparador B conmuta de nivel BAJO a nivel ALTO y pone en estado SET al latch S-R, haciendo que la salida (pin 3) pase a nivel ALTO y bloqueando el transistor de descarga Q_1. La salida permanecerá a nivel ALTO hasta que la tensión umbral, normalmente a nivel BAJO sobrepase *2/3 de* V_{CC} y haga que la salida del comparador A conmute de nivel BAJO a nivel ALTO. Esto hace que el latch pase a estado RESET, con lo que la salida se pone de nuevo a nivel BAJO, de manera que el transistor de descarga se activa. La entrada de puesta a cero (RESET) externa se puede utilizar para poner el latch a cero, independientemente del circuito umbral. Las entradas de disparo y umbral (pines 2 y 6) se controlan mediante componentes externos, para establecer el modo de funcionamiento como monoestable o aestable.

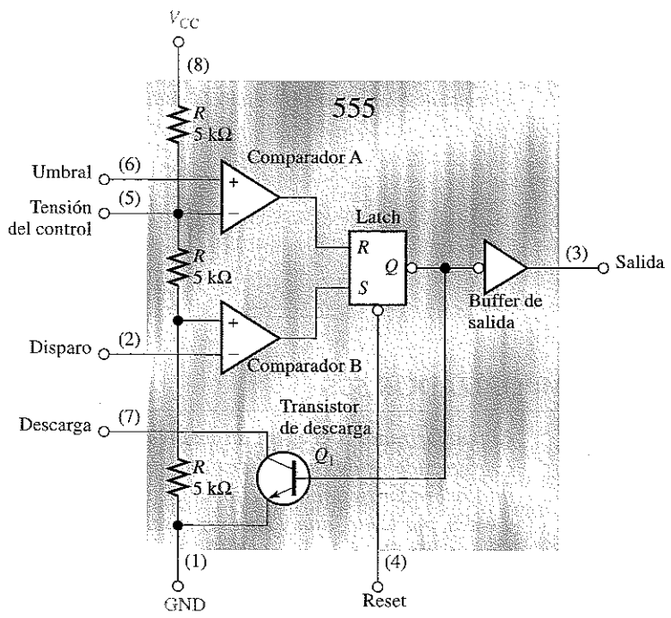


FIGURA 8.56
Diagrama funcional interno de un temporizador 555 (la numeración de pines se indica entre paréntesis).

Funcionamiento como monoestable

Para configurar un temporizador 555 como monoestable no redisparable, se utilizan una resistencia y un condensador externos, tal como se muestra en la Figura 8.57. La anchura del impulso de salida se determina mediante la constante de tiempo, que se calcula a partir de R_1 y C_1 según la siguiente fórmula:

$$t_W = 1,1\ R_1\ C_1 \qquad \textbf{(8.3)}$$

La entrada de la tensión de control no se utiliza y se conecta a un condensador de desacoplo C_2, para evitar la aparición de ruido que pudiera afectar los niveles umbral y de disparo.

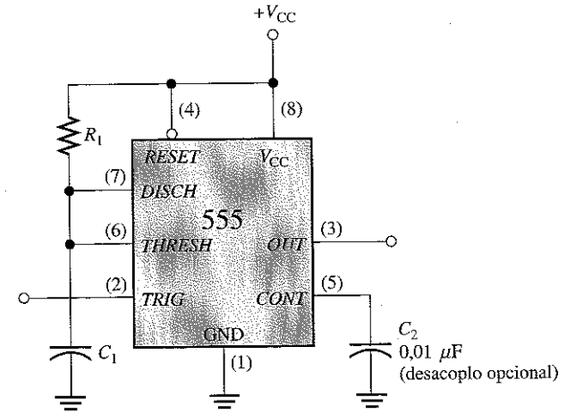


FIGURA 8.57
El temporizador 555 conectado como monoestable.

Antes de aplicar el impulso de disparo, la salida está a nivel BAJO y el transistor de descarga Q_1 conduce, como se observa en la Figura 8.58(a). Cuando se aplica un impulso de disparo negativo, la salida pasa a nivel ALTO y el transistor de descarga se bloquea, permitiendo al condensador C_1 comenzar a cargarse a través de R_1, como se muestra en la parte (b). Cuando C_1 se ha cargado hasta *1/3 de* V_{CC}, la salida pasa de nuevo a nivel BAJO y Q_1 entra en conducción inmediatamente, descargándose C_1, como se indica en la parte (c). Como puede verse, la velocidad de carga de C_1 determina cuánto tiempo va a estar la salida a nivel ALTO.

EJEMPLO 8.15 ¿Cuál es la anchura del impulso de salida para un circuito monoestable 555 con R_1 = 2,2 kΩ y C_1 = 0,01 μF?

Solución

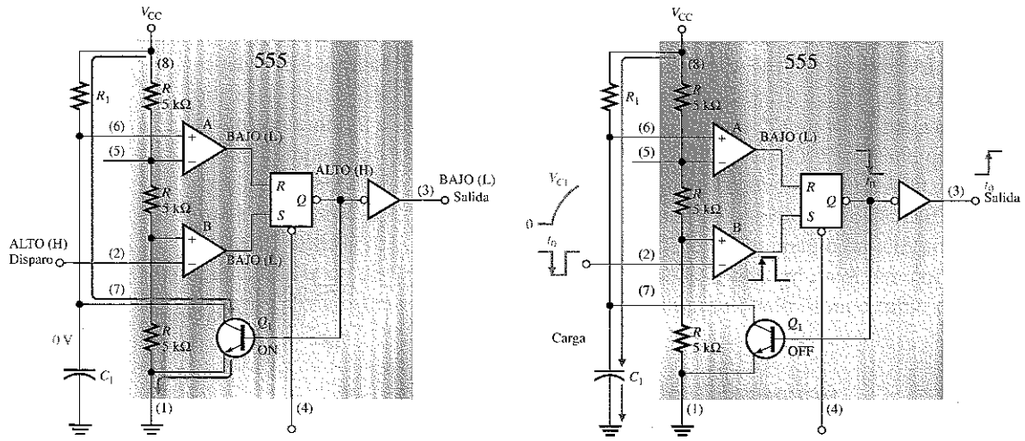
A partir de la ecuación (8.3), la anchura del impulso es:

$t_W = 1,1\ R_1\ C_1 = 1,1\ (2,2\ \text{k}\Omega)\ (0,01\ \mu\text{F}) =$ **24,2 μs**

Problema relacionado. Para $C_1 = 0{,}01\ \mu F$, determinar el valor de R_1 para una anchura de impulso de 1 ms.

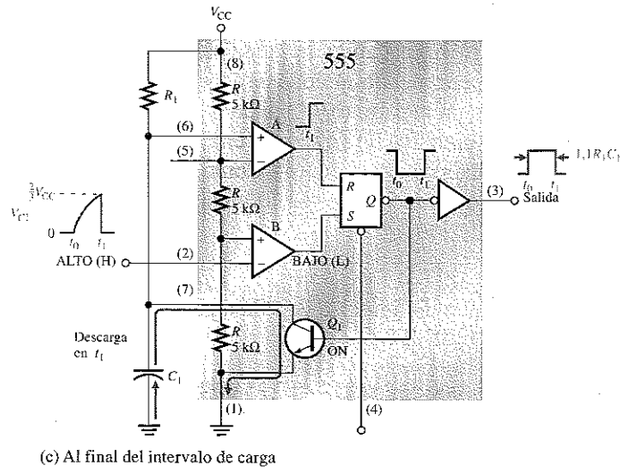
Funcionamiento como aestable

En la Figura 8.59 se muestra un temporizador 555 conectado para funcionar como multivibrador **aestable**, que es un **oscilador** no sinusoidal. Obsérvese que, en este caso, la entrada umbral (*THRESH*) está conectada a la entrada de disparo (*TRIG*). Los componentes externos R_1, R_2 y C_1 conforman la red de temporización que determina la frecuencia de oscilación. El condensador C_2 de 0,01 μF conectado a la entrada de control (*CONT*) sirve únicamente para desacoplar y no afecta en absoluto al funcionamiento del resto del circuito; en algunos casos se puede eliminar.



(a) Antes del disparo

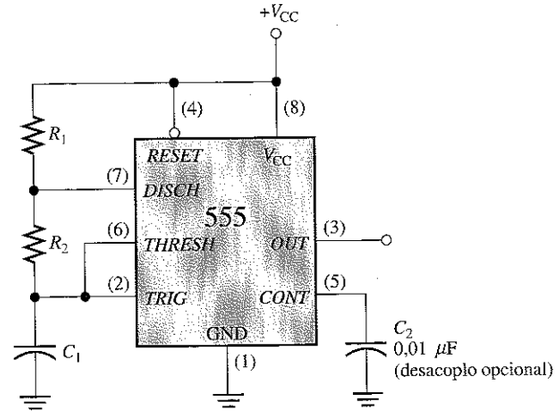
(b) Durante el disparo



(c) Al final del intervalo de carga

FIGURA 8.58
Funcionamiento del temporizador 555 configurado como monoestable.

FIGURA 8.59
El temporizador 555 configurado como multivibrador aestable (oscilador).



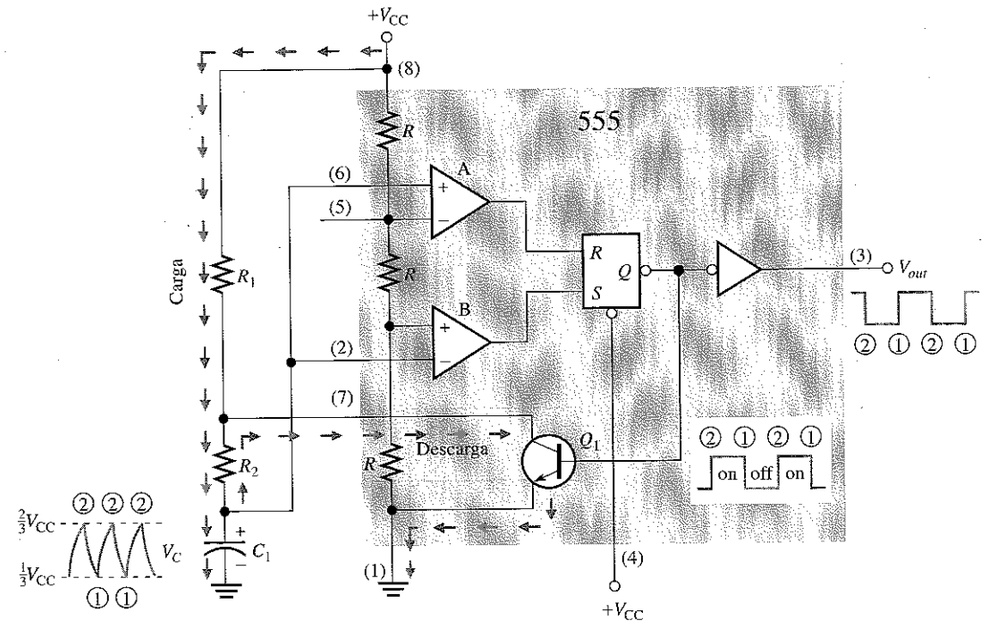


FIGURA 8.60
Funcionamiento del temporizador 555 configurado como aestable.

Inicialmente, cuando se conecta la alimentación, el condensador está descargado y, por tanto, la tensión de disparo (pin 2) es 0 V. Esto da lugar a que la salida del comparador B esté a nivel ALTO y la salida del comparador A a nivel BAJO, forzando la salida del latch, y por consiguiente la base de Q_1 a nivel BAJO, manteniendo el transistor bloqueado. A continuación, C_1 comienza a cargarse a través de R_1 y R_2, tal como se indica en la Figura 8.60. Cuando la tensión del condensador alcanza el valor

NOTAS INFORMÁTICAS
Todas las computadoras requieren una fuente de temporización para proporcionar señales de reloj precisas. La sección de temporización controla todas las temporizaciones del sistema y es responsable del correcto funcionamiento del hardware del sistema. Normalmente, la sección de temporización consta de un oscilador controlado de cristal y de contadores para realizar la división de frecuencia. El uso de un oscilador de alta frecuencia y de divisores para obtener una frecuencia menor proporciona una mayor precisión y estabilidad en frecuencia.

de 1/3 V_{CC}, el comparador B cambia a su nivel de salida BAJO, y cuando la tensión del condensador alcanza el valor de 2/3 V_{CC}, el comparador A cambia a su nivel de salida ALTO. Esto pone en estado de RESET al latch, haciendo que la base de Q_1 pase a nivel ALTO, activando el transistor. Esta secuencia origina un camino de descarga para el condensador a través de R_2 y del transistor, tal como se indica. El condensador comienza ahora a descargarse, haciendo que el comparador A pase a nivel BAJO. En el momento en que el condensador se descarga hasta el valor 1/3 V_{CC}, el comparador B conmuta a nivel ALTO, poniendo al latch en estado SET, lo que hace que la base de Q_1 se ponga a nivel BAJO, bloqueando el transistor. De nuevo comienza otro ciclo de carga, y el proceso completo se repite. El resultado es una señal de salida rectangular cuyo ciclo de trabajo depende de los valores de R_1 y R_2. La frecuencia de oscilación viene dada por la siguiente fórmula, o puede también hallarse utilizando el gráfico de la Figura 8.61.

$$f = \frac{1{,}44}{(R_1 + 2R_2)C_1} \qquad \textbf{(8.4)}$$

El ciclo de trabajo de la salida puede ser ajustado seleccionando R_1 y R_2. Dado que C_1 se carga a través de $R_1 + R_2$ y se descarga únicamente a través de R_2, se pueden conseguir ciclos de trabajo de un mínimo del 50 por ciento aproximadamente, si $R_2 >> R_1$, de forma que los tiempos de carga y descarga sean aproximadamente iguales.

La expresión para el ciclo de trabajo se obtiene de la manera siguiente. El intervalo de tiempo en que la salida está a nivel ALTO (t_H) representa lo que tarda C_1 en cargarse desde 1/3 V_{CC} hasta 2/3 V_{CC}. Esto se expresa como:

$$t_H = 0{,}7(R_1 + R_2)C_1 \qquad \textbf{(8.5)}$$

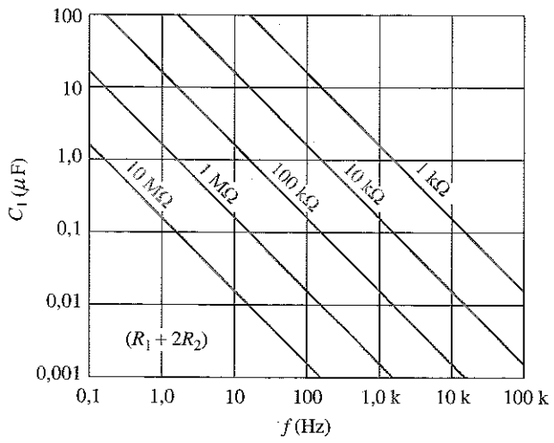


FIGURA 8.61
Frecuencia de oscilación en función de C_1 y $R_1 + 2R_2$. Las líneas diagonales representan los valores de $R_1 + 2R_2$.

El intervalo de tiempo durante el que la salida está a nivel BAJO (t_L) representa lo que tarda C_1 en descargarse desde $1/3V_{CC}$ hasta $2/3V_{CC}$. Esto se expresa como:

$$t_L = 0{,}7R_2C_1 \qquad \textbf{(8.6)}$$

El periodo, T, de la señal de salida es la suma de t_H y t_L:

$$T = t_H + t_L = 0{,}7\,(R_1 + 2R_2)\,C_1$$

Esto es el recíproco de f en la Ecuación (8.4). Finalmente, el ciclo de trabajo es:

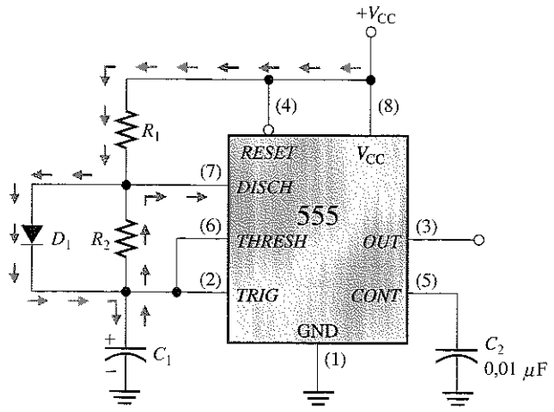
$$\text{Ciclo de trabajo} = \frac{t_H}{T} = \frac{t_H}{t_H + t_L}$$

$$\text{Ciclo de trabajo} = \left(\frac{R_1 + R_2}{R_1 + 2R_2}\right)100\% \qquad \textbf{(8.7)}$$

Para conseguir ciclos de trabajo menores que el 50 por ciento, se puede modificar el circuito de la Figura 8.59, de modo que C_1 se cargue sólo a través de R_1 y se descargue a través de R_2. Esto se consigue mediante un diodo D_1 colocado tal y como se muestra en la Figura 8.62. El ciclo de trabajo se puede hacer menor que el 50 por ciento, haciendo R_1 menor que R_2. Bajo esta condición, la expresión para el ciclo de trabajo es:

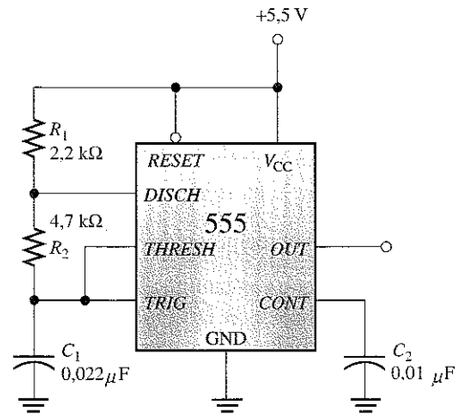
$$\text{Ciclo de trabajo} = \left(\frac{R_1}{R_1 + R_2}\right)100\% \qquad \textbf{(8.8)}$$

FIGURA 8.62
La adición de un diodo D_1 permite ajustar el ciclo de trabajo de la salida a un valor menor del 50 por ciento, haciendo $R_1 < R_2$.



EJEMPLO 8.16 En la Figura 8.63 se muestra un temporizador 555 configurado para funcionar en modo aestable (oscilador). Determinar la frecuencia de la salida y el ciclo de trabajo.

FIGURA 8.63



Solución. Utilizar las Ecuaciones (8.4) y (8.7).

$$f = \frac{1{,}44}{(R_1 + 2R_2)C_1} = \frac{1{,}44}{(2{,}2\ \text{k}\Omega + 9{,}4\ \text{k}\Omega)0{,}022\ \mu\text{F}} = \mathbf{5{,}64\ kHz}$$

$$\text{Ciclo de trabajo} = \left(\frac{R_1 + R_2}{R_1 + 2R_2}\right) 100\% = \left(\frac{2{,}2\,\text{k}\Omega + 4{,}7\,\text{k}\Omega}{2{,}2\,\text{k}\,\Omega + 9{,}4\,\text{k}\,\Omega}\right) 100\% = \mathbf{59{,}5\%}$$

Problema relacionado. Determinar el ciclo de trabajo en la Figura 8.63 si se conecta un diodo en paralelo con R_2, como se indica en la Figura 8.62.

REVISIÓN DE LA SECCIÓN 8.7

1. Explicar la diferencia de funcionamiento entre un multivibrador aestable y un multivibrador monoestable.
2. Para un determinado multivibrador aestable, $t_H = 15$ ms y $T = 20$ ms. ¿Cuál es el ciclo de trabajo de la salida?

8.8 ■ LOCALIZACIÓN DE AVERÍAS

En la industria, es una práctica habitual probar los nuevos diseños de circuitos para asegurarse de que funcionan como se ha especificado. Usualmente, se realiza un montaje provisional de los nuevos diseños y se prueban antes de que finalice el diseño. El término montaje provisional se refiere a la forma en que el nuevo circuito se ensambla para poder verificar su funcionamiento y, en caso de encontrarse fallos, eliminarlos antes de que se construya el prototipo. En esta sección, vamos a considerar un ejemplo de esto.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir cómo la temporización de un circuito puede producir glitches. ■ Afrontar la depuración de un nuevo diseño con una mayor comprensión y conciencia de los problemas potenciales.

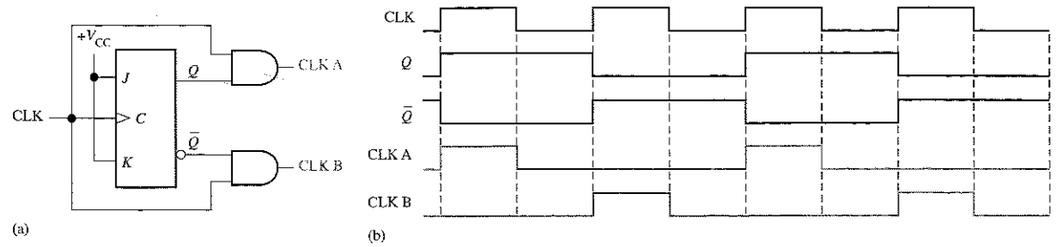
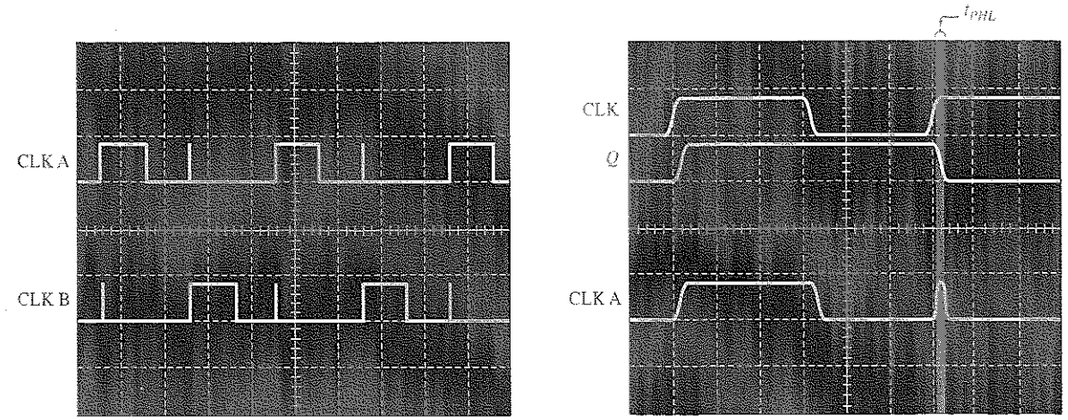


FIGURA 8.64
Generador de reloj de dos fases con señales ideales.

El circuito que se muestra en la Figura 8.64(a) genera dos señales de reloj (CLK A y CLK B), siendo la aparición de los impulsos alternativa. Cada señal tiene la frecuencia mitad de la señal de reloj original (CLK), como se muestra en el diagrama de tiempos ideal de la parte (b).

Cuando se prueba el circuito, las señales CLK A y CLK B se presentan en el osciloscopio o en el analizador lógico como se muestra en la Figura 8.65(a). Puesto que se observan glitches en ambas señales, deducimos que existe algún fallo en el circuito, bien en su diseño básico o en la manera en que está conectado. Investigaciones posteriores revelan que los glitches son causados por una condición de **concurrencia** entre la señal CLK y las señales Q y $\bar{Q}$ en las entradas de las puertas AND. Como se muestra en la Figura 8.65(b), los retardos de propagación entre CLK y Q y $\bar{Q}$ dan lugar a una coincidencia de corta duración de niveles altos en los flancos anteriores de los impulsos alternos de reloj. Por tanto, existe un error de diseño.



(a) Pantalla del analizador lógico que muestra las señales CLK A y CLK B indicando los glitches mediante "picos".

(b) Pantalla del analizador lógico que muestra el retardo de propagación que origina un glitch en la señal CLK A

FIGURA 8.65
Pantallas del analizador lógico para el circuito de la Figura 8.64.

El problema puede ser corregido mediante el uso de un flip-flop disparado por flanco negativo en lugar del dispositivo disparado por flanco positivo, como se observa en la Figura 8.66(a). Aunque el retraso en la propagación entre CLK y Q y $\bar{Q}$ sigue existiendo, éstos se inician en los flancos posteriores del reloj (CLK), eliminando por tanto los glitches, como se muestra en el diagrama de tiempos de la Figura 8.66(b).

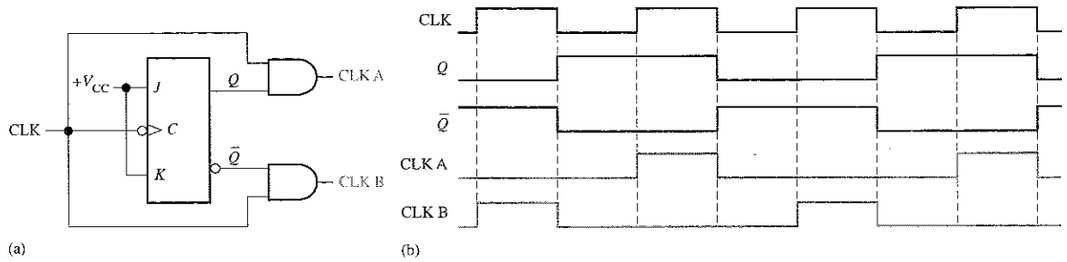


FIGURA 8.66
Generador de reloj de dos fases, que utiliza flip-flops disparados por flanco negativo para eliminar los glitches.

Consejos Prácticos

Los glitches que se producen en los sistemas digitales son muy rápidos (extremadamente cortos en duración) y pueden ser difíciles de ver en un osciloscopio, especialmente a muy bajas velocidades de barrido. Sin embargo, un analizador lógico puede mostrar un glitch fácilmente. Para localizar glitches utilizando un analizador lógico, debe seleccionarse el modo "latch" o el muestreo transicional, si está disponible. En el modo latch, el analizador busca un cambio de nivel de tensión. Cuando se produce un cambio, incluso aunque sea de una duración extremadamente corta (unos pocos nanosegundos), la información se almacena en la memoria del analizador como cualquier otro dato muestreado. Cuando se muestran los datos, el glitch se presentará como un cambio obvio en los datos muestreados, por lo que se puede identificar fácilmente.

REVISIÓN DE LA SECCIÓN 8.8

1. ¿Se puede usar un flip-flop D disparado por flanco negativo en el circuito de la Figura 8.66?
2. ¿Qué dispositivo puede utilizarse como reloj en el circuito de la Figura 8.66?

8.9 ■ APLICACIÓN A LOS SISTEMAS DIGITALES

En esta aplicación a los sistemas digitales, vamos a continuar trabajando con el sistema de control de semáforos que se inició en el Capítulo 6. Conviene repasar la sección 6.12 antes de continuar. En este capítulo, el punto de interés se centrará en los circuitos de temporización que generan los intervalos de tiempo de 4 segundos para la luz ámbar y de 25 segundos para las luces roja y verde, así como el reloj.

Al finalizar esta sección, el lector deberá ser capaz de:

- Desarrollar un diagrama de bloques para los circuitos de temporización del sistema.
- Implementar los temporizadores corto y largo que cumplen las especificaciones.
- Implementar el oscilador del reloj que cumple las especificaciones. ■ Desarrollar un esquema a partir de la tarjeta de circuito impreso. ■ Desarrollar un procedimiento de pruebas y localización de averías para la tarjeta del circuito.

Requerimientos generales de los circuitos de temporización

El diagrama de bloques general se estudió en la aplicación del Capítulo 6, aunque se muestra de nuevo en la Figura 8.67 como referencia. Los bloques para la decodificación de estados y la lógica de salida ya han sido desarrollados, y ahora nos vamos a concentrar en los circuitos de temporización que deben generar un impulso positivo de 4 segundos o un impulso positivo de 25 segundos al ser disparados por el decodificador de estados y la lógica de salida, así como la generación de la señal de reloj de 10 kHz. Estas tres salidas las utiliza la lógica secuencial que será desarrollada en el Capítulo 9.

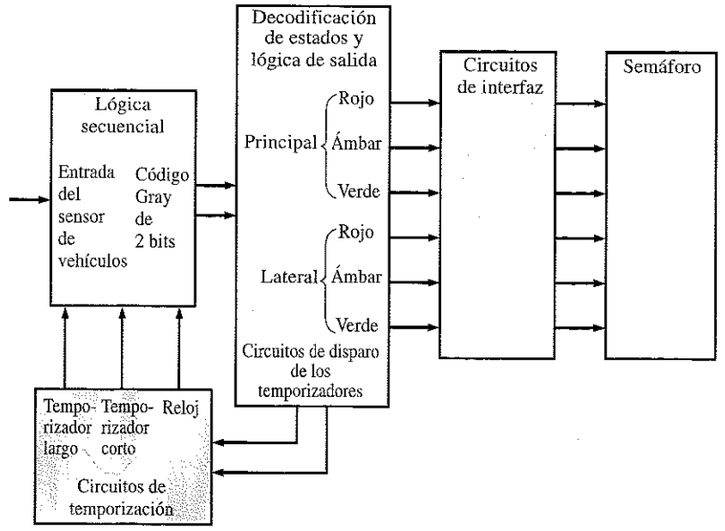


FIGURA 8.67
Diagrama de bloques del sistema de control de semáforos.

Los circuitos temporizadores están formados por el temporizador largo, el temporizador corto y el oscilador del reloj, como muestra la Figura 8.68. El temporizador largo y el corto se implementan mediante monoestables, mientras que el oscilador del reloj se implementa a partir de un temporizador 555 configurado en modo aestable.

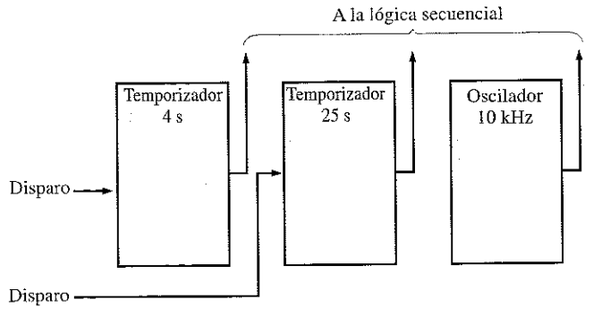


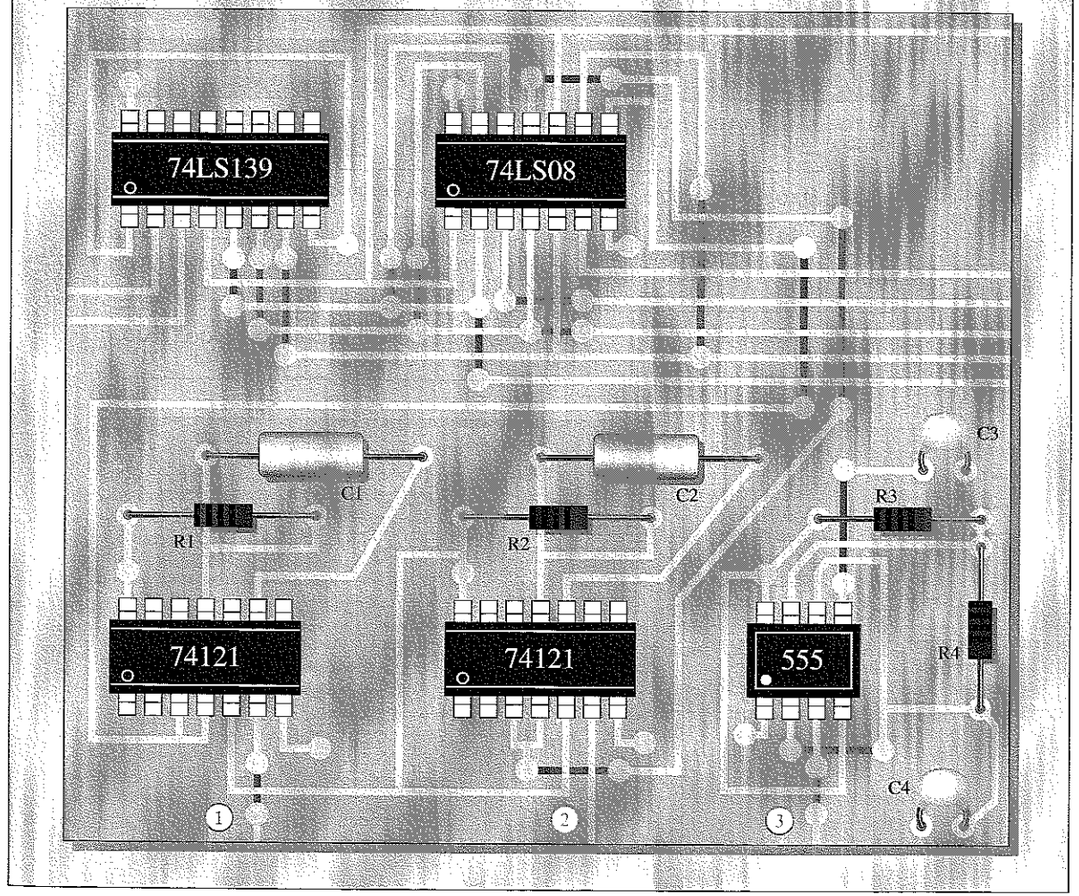
FIGURA 8.68
Diagrama de bloques de los circuitos de temporización.

■ PRÁCTICAS DE SISTEMAS

En la aplicación a sistemas digitales del Capítulo 6, nos centramos en la parte de la tarjeta controladora del semáforo que contenía el decodificador de estados y la lógica de salida. En esta aplicación, nos centraremos en la parte correspondiente a los circuitos de temporización. Como antes, todas las conexiones a V_{CC} y a masa se encuentran en la parte posterior de la tarjeta. Las interconexiones del circuito sobre la cara posterior de la tarjeta se hacen a través de los taladros de la cara de componentes.

■ PRÁCTICA DE SISTEMAS 1: Diseño y verificación del circuito

- ***Actividad 1.*** Utilizando la tarjeta de circuito impreso (PC board), añadir la parte del circuito de temporización del sistema al esquemático que se comenzó en la Sección 6.11.
- ***Actividad 2.*** Verificar que el circuito de temporización de la tarjeta es correcto y está apropiadamente conectado al decodificador de estados y a la lógica de salida.
- ***Actividad 3.*** Identificar la función de cada uno de los puntos de la tarjeta situados a la derecha de cada uno de los tres círculos numerados en la parte inferior de la tarjeta.
- ***Actividad 4.*** Determinar el valor de cada resistencia y condensador de la tarjeta que producirán un impulso positivo de 4 segundos, un impulso positivo de 25 segundos y un tren de ondas a 10 kHz (el ciclo de trabajo no es crítico).

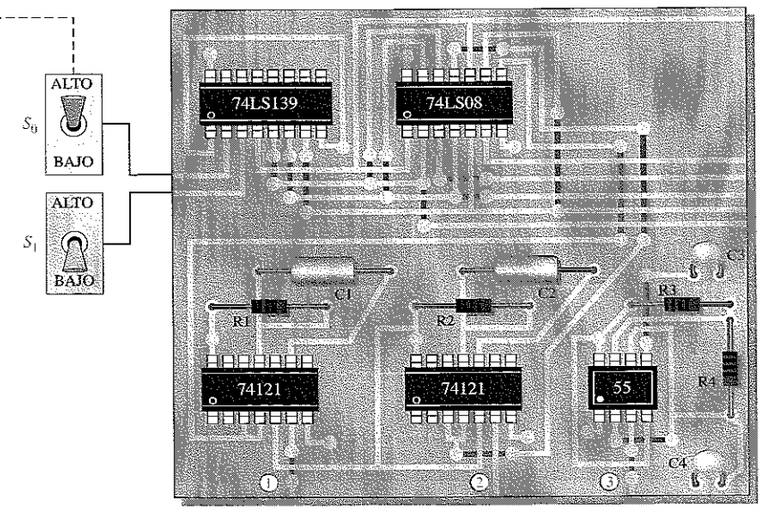


■ PRÁCTICA DE SISTEMAS 2: Pruebas y localización de averías

- ***Actividad 1.*** Utilizando los conmutadores conectados tal y como se muestra para simular entradas en código Gray procedentes de la etapa secuencial del sistema, desarrollar un procedimiento de pruebas para verificar esta parte del sistema.
- ***Actividad 2.*** Si se colocan los conmutadores tal y como se muestra en la tabla, determinar a partir de las indicaciones en los puntos de prueba representados por los círculos numerados si el circuito está funcionando adecuadamente en cada caso y, si no es así, cuál puede ser la causa más probable del fallo.

La lógica de la tarjeta de circuito impreso para la decodificación de estados, la lógica de salida y los circuitos de temporización se encuentra en el archivo SA08.EWB del CD-ROM EWB. Utilizar EWB para simular una prueba de este circuito.

S_1	S_0	Caso 1 ①	Caso 1 ②	Caso 1 ③	Caso 2 ①	Caso 2 ②	Caso 2 ③	Caso 3 ①	Caso 3 ②	Caso 3 ③	Caso 4 ①	Caso 4 ②	Caso 4 ③
ALTO	ALTO	ALTO (H) 25 s	BAJO (L)	10 kHz	BAJO (L)	ALTO (H) 25 s	10 kHz	ALTO (H) 25 s	BAJO (L)	10 kHz	BAJO (L)	BAJO (L)	BAJO (L)
BAJO	ALTO	BAJO (L)	ALTO (H) 4 s	10 kHz	ALTO (H) 4 s	BAJO (L)	10 kHz	BAJO (L)	BAJO (L)	10 kHz	BAJO (L)	BAJO (L)	BAJO (L)
ALTO	BAJO	ALTO (H) 25 s	BAJO (L)	10 kHz	BAJO (L)	ALTO (H) 25 s	10 kHz	ALTO (H) 25 s	BAJO (L)	10 kHz	BAJO (L)	BAJO (L)	BAJO (L)
BAJO	BAJO	BAJO (L)	ALTO (H) 4 s	10 kHz	ALTO (H) 4 s	BAJO (L)	10 kHz	BAJO (L)	BAJO (L)	10 kHz	BAJO (L)	BAJO (L)	BAJO (L)



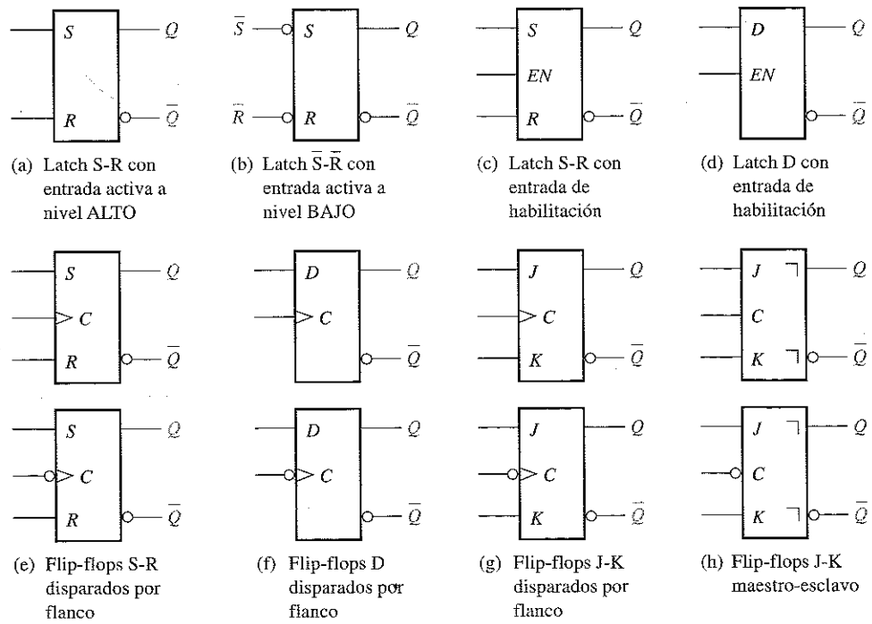
REVISIÓN DE LA SECCIÓN 8.9

1. Explicar el propósito de los dos conmutadores del banco de trabajo 2.
2. ¿Por qué se disparan por separado los monoestables de 4 y de 25 segundos?
3. ¿De qué otras partes del sistema depende el funcionamiento del oscilador del reloj?

En este punto se puede estudiar el Capítulo 15, "Tecnología de circuitos integrados", parcial o totalmente. El examen de este capítulo no es un prerrequisito para abordar cualquier otro tema tratado en este texto, y su omisión total o parcial no afectará al resto de los temas.

■ RESUMEN

- Los símbolos de los latches y flip-flops se muestran en la Figura 8.69.
- Los latches son dispositivos biestables cuyo estado depende normalmente de entradas asíncronas.
- Los flip-flops disparados por flanco son elementos de dos estados con entradas síncronas, cuyo estado depende de las entradas sólo durante la transición de disparo de un impulso de reloj. Los cambios en las salidas ocurren durante las transiciones de disparo del reloj.



(a) Latch S-R con entrada activa a nivel ALTO
(b) Latch $\bar{S}$-$\bar{R}$ con entrada activa a nivel BAJO
(c) Latch S-R con entrada de habilitación
(d) Latch D con entrada de habilitación
(e) Flip-flops S-R disparados por flanco
(f) Flip-flops D disparados por flanco
(g) Flip-flops J-K disparados por flanco
(h) Flip-flops J-K maestro-esclavo

FIGURA 8.69

- Los flip-flops maestro-esclavo disparados por pulsos son elementos de dos estados con entradas síncronas, cuyo estado depende de las entradas durante los flancos anteriores de los impulsos del reloj, pero cuya salida se encuentra pospuesta y no refleja el estado interno hasta que llega el flanco posterior de los impulsos de reloj. No debe permitirse a las entradas síncronas cambiar mientras que el reloj se encuentre en estado ALTO.

- Los multivibradores monoestables tienen un único estado estable. Cuando se dispara un monoestable, la salida pasa a su estado inestable durante un tiempo que está determinado por un circuito *RC*.
- Los multivibradores aestables no poseen ningún estado estable y se utilizan como osciladores para generar señales de temporización en los sistemas digitales.

■ PALABRAS CLAVE

Estos términos también se encuentran en el glosario del final del libro.

Aestable Que no tiene ningún estado estable. Un multivibrador aestable oscila entre dos estados semiestables.

Asíncrono Que no tiene ninguna relación temporal fija.

Basculación Acción de un flip-flop cuando cambia de estado con cada impulso de reloj.

Biestable Que tiene dos estados estables. Los flip-flops y los latches son multivibradores biestables.

Borrado (clear) Entrada asíncrona utilizada para resetear un flip-flop. Acción de poner un registro o contador en el estado en que contiene solamente ceros.

Flip-flop D Un tipo de multivibrador biestable en el que la salida sigue al estado de la entrada D.

Flip-flop disparado por flanco Un tipo de flip-flop en el que los datos se introducen y aparecen en la salida durante el mismo flanco del impulso del reloj.

Flip-flop J-K Un tipo de flip-flop que puede funcionar en los modos de SET, RESET, no cambio y basculación.

Flip-flop maestro-esclavo Un tipo de flip-flop en que los datos de entrada se introducen en el dispositivo durante el flanco anterior del pulso de reloj y aparecen en la salida durante el flanco posterior.

Flip-flop S-R Flip-flop SET-RESET.

Histéresis Una característica de los circuitos con umbral de disparo, como el trigger Schmitt, donde el dispositivo se activa y desactiva para diferentes niveles de entrada.

Inicialización (preset) Entrada asíncrona para inicializar un flip-flop.

Latch Dispositivo digital biestable utilizado para almacenar un bit.

Monoestable Que tiene un solo estado estable. Un multivibrador monoestable, o sencillamente un monoestable, produce un único impulso en respuesta a una entrada de disparo.

RESET Estado de un flip-flop o latch cuando la salida es 0. La acción de producir un estado de desactivación.

SET Estado de un flip-flop o latch cuando la salida es 1. La acción de producir un estado de activación.

Síncrono Que tiene una relación temporal fija.

Temporizador Circuito que puede ser utilizado como monoestable o como oscilador.

Tiempo de set-up (establecimiento) Intervalo de tiempo que los niveles de las señales de entrada deben mantenerse en las entradas de un circuito digital, como puede ser un flip-flop, antes del flanco de disparo del impulso de reloj.

Tiempo de hold (mantenimiento) El intervalo de tiempo que los niveles de las señales de entrada deben permanecer en las entradas de un flip-flop después del flanco de disparo del reloj, de manera que se active fiablemente el dispositivo.

■ AUTOTEST

1. Si un latch S-R tiene un 1 en su entrada S y un 0 en su entrada R, pasando posteriormente su entrada S a 0, el latch estará:
   (a) SET **(b)** RESET **(c)** condición no válida **(d)** borrado
2. El estado no válido de un latch S-R ocurre cuando:
   (a) $S = 1, R = 0$ **(b)** $S = 0, R = 1$ **(c)** $S = 1, R = 1$ **(d)** $S = 0, R = 0$
3. En un latch D con entrada de habilitación, la salida Q siempre coincide con la entrada D
   (a) antes del impulso de habilitación
   (b) durante el impulso de habilitación
   (c) inmediatamente después del impulso de habilitación
   (d) (b) y (c)
4. Al igual que los latch, los flip-flops pertenecen a una categoría de circuitos lógicos conocidos como:
   (a) multivibradores monoestables **(b)** multivibradores biestables
   (c) multivibradores aestables **(d)** monoestables
5. La función de la entrada de reloj en un flip-flop es para:
   (a) borrar el dispositivo
   (b) activar (SET) el dispositivo
   (c) obligar siempre a la salida a cambiar de estado
   (d) obligar a la salida a asumir un estado que depende de las entradas de control (*S-R, J-K* o *D*)
6. En un flip-flop D disparado por flanco
   (a) sólo puede ocurrir un cambio en el estado del flip-flop durante algún flanco del impulso del reloj
   (b) el estado al que pasa el flip-flop depende de la entrada D
   (c) la salida toma el mismo valor de la entrada en cada impulso del reloj
   (d) las tres respuestas anteriores
7. Una característica que distingue un flip-flop J-K de uno S-R es
   (a) la condición de basculación **(b)** la entrada de inicialización
   (c) el tipo de reloj **(d)** la entrada de borrado
8. Un flip-flop J-K se encuentra en modo de basculación cuando
   (a) $J = 1, K = 0$ **(b)** $J = 1, K = 1$
   (c) $J = 0, K = 0$ **(d)** $J = 0, K = 1$
9. Un flip-flop J-K con $J = 1$ y $K = 1$ tiene una entrada de reloj de 10 kHz. La salida Q es
   (a) constantemente un nivel *ALTO* **(b)** constantemente un nivel *BAJO*
   (c) una señal cuadrada de 10 kHz **(d)** una señal cuadrada de 5 kHz
10. Un monoestable es un tipo de
    (a) multivibrador con un único estado estable
    (b) multivibrador aestable
    (c) temporizador
    (d) (a) y (c)
    (e) (b) y (c)

11. La anchura del impulso de salida de un monoestable no redisparable depende de
- **(a)** los intervalos de disparo
- **(b)** la alimentación
- **(c)** de una resistencia y un condensador
- **(d)** la tensión umbral

12. Un multivibrador aestable
- **(a)** requiere una entrada periódica de disparo
- **(b)** no tiene ningún estado estable
- **(c)** es un oscilador
- **(d)** produce un impulso de salida periódico
- **(e)** (a), (b), (c) y (d)
- **(f)** (b), (c) y (d)

■ PROBLEMAS SECCIÓN 8.1 Latches

1. Si se aplican las señales de la Figura 8.70 a un latch S-R con entradas activas a nivel BAJO, dibujar la forma de onda de salida Q resultante en función de las entradas. Suponer que, inicialmente, Q está a nivel BAJO.

2. Resolver el problema 1 para las formas de onda de entrada de la Figura 8.71, aplicadas a un latch S-R activo a nivel ALTO.

3. Resolver el problema 1 para las formas de onda de entrada de la Figura 8.72.

FIGURA 8.70

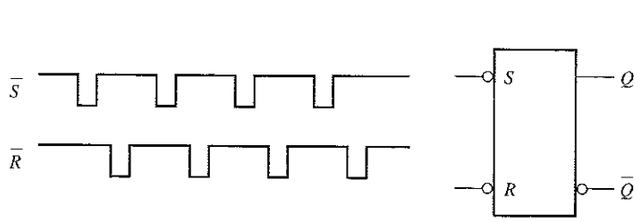


FIGURA 8.71

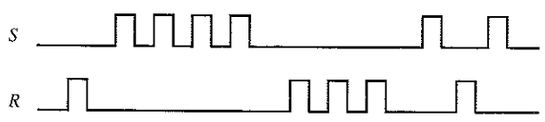
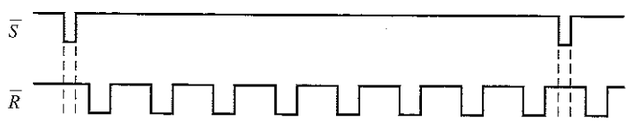


FIGURA 8.72



4. Determinar las salidas Q y $\overline{Q}$ de un latch S-R con entrada de habilitación para las entradas de la Figura 8.73. Dibujarlas en función de la entrada de habilitación. Suponer que, inicialmente, Q está a nivel BAJO.

5. Resolver el problema 4 para las entradas de la Figura 8.74.

6. Resolver el problema 4 para las entradas de la Figura 8.75.

7. En un latch D con entrada de habilitación, se observan en sus entradas las formas de onda de la Figura 8.76. Dibujar el diagrama de tiempos, mostrando la forma de onda de salida que esperaríamos observar en Q si el latch se encuentra inicialmente en estado RESET.

FIGURAS 8.73

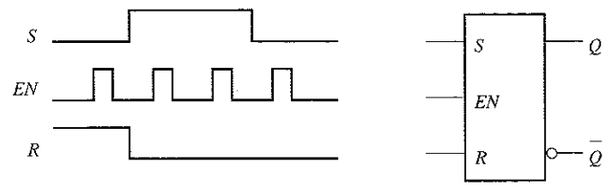


FIGURA 8.74

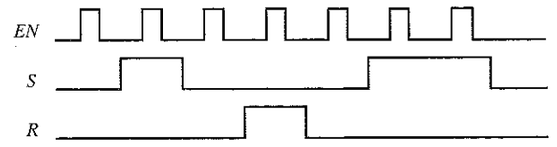


FIGURA 8.75

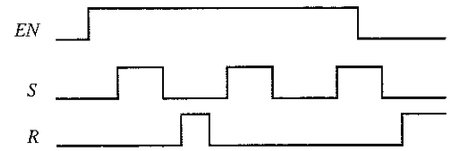
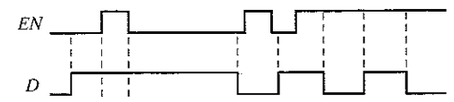


FIGURA 8.76



SECCIÓN 8.2 Flip-flops disparados por flanco

8. En la Figura 8.77 se muestran dos flip-flops S-R disparados por flanco. Si las entradas son las que se indican, dibujar la salida Q de cada flip-flop en función de la señal de reloj y explicar la diferencia entre los dos. Los flip-flops se encuentran inicialmente en estado RESET.

9. La salida Q de un flip-flop S-R disparado por flanco se muestra en la Figura 8.78 en función de la señal del reloj. Determinar las formas de onda de entrada que se necesitan en las entradas S y R para producir esta salida, si el flip-flop es de tipo disparado por flanco positivo.

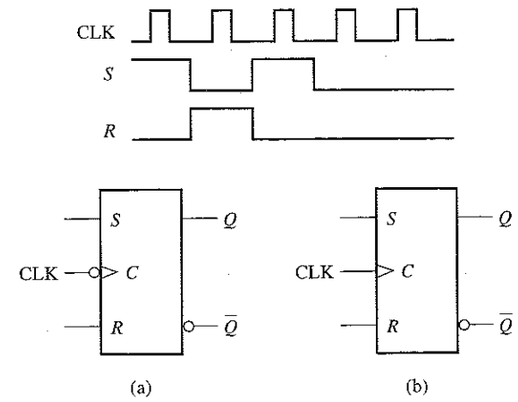
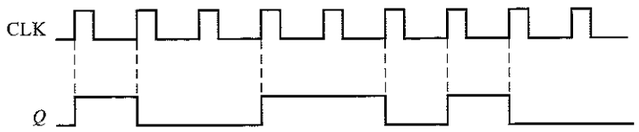


FIGURA 8.77

FIGURA 8.78



10. Dibujar la salida Q en función del reloj para un flip-flop D cuyas entradas son las que se muestran en la Figura 8.79. Suponer disparo por flanco positivo y que Q se encuentra inicialmente a nivel BAJO.

11. Resolver el problema 10 para las entradas de la Figura 8.80.

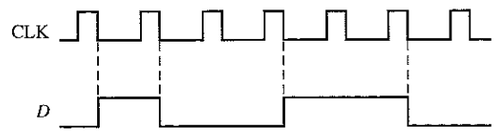


FIGURA 8.79

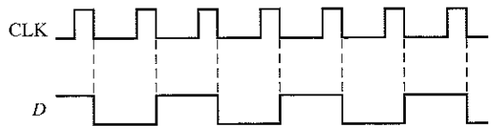


FIGURA 8.80

12. Para un flip-flop J-K disparado por flanco positivo cuyas entradas son las que se muestran en la Figura 8.81, determinar la salida Q en función del reloj. Suponer que, inicialmente, Q está a nivel BAJO.

13. Resolver el problema 12 para las entradas de la Figura 8.82.

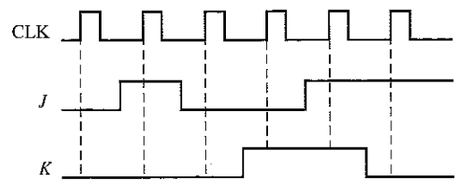


FIGURA 8.81

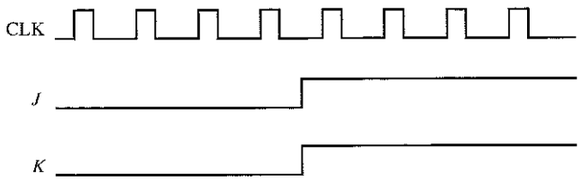


FIGURA 8.82

14. Determinar la salida Q en función del reloj si las señales que se muestran en la Figura 8.83 se aplican a las entradas de un flip-flop J-K. Suponer que Q se encuentra inicialmente a nivel BAJO.

15. Para un flip-flop J-K disparado por flanco negativo cuyas entradas son las de la Figura 8.84, dibujar la forma de onda de salida Q en función del reloj. Suponer que Q se encuentra inicialmente a nivel BAJO.

FIGURA 8.83

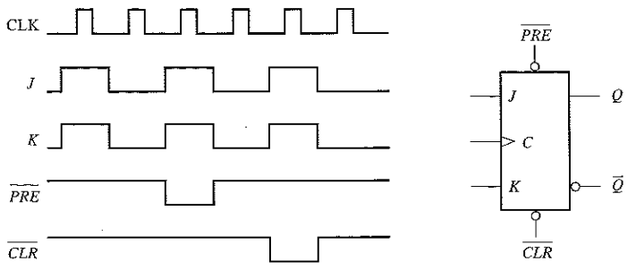
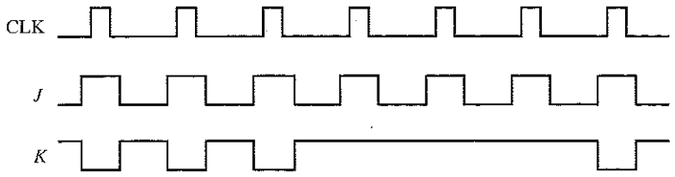


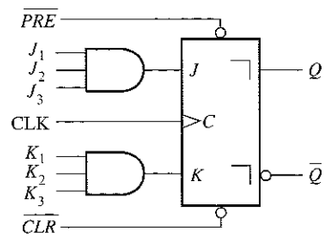
FIGURA 8.84



16. Se aplican los siguientes datos serie a un flip-flop a través de puertas AND, como se indica en la Figura 8.85. Determinar los datos serie resultantes que aparecen en la salida Q. Hay un impulso de reloj por cada periodo de bit. Suponer que, inicialmente, Q es 0 y, $\overline{PRE}$ y $\overline{CLR}$ están a nivel ALTO. Los bits de más a la derecha son los primeros que se aplican.

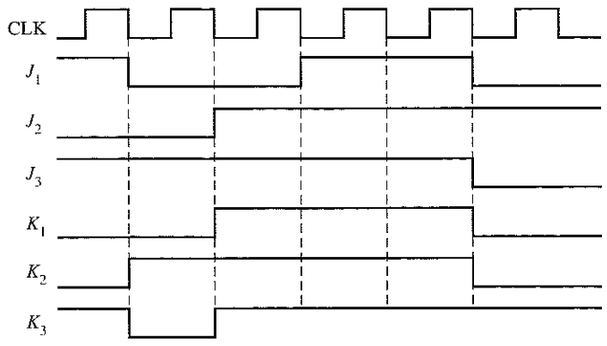
J_1: 1010011
J_2: 0111010
J_3: 1111000
K_1: 0001110
K_2: 1101100
K_3: 1010101

FIGURA 8.85



17. Completar el diagrama de tiempos de la Figura 8.86 para el circuito de la Figura 8.85, dibujando la salida Q que, inicialmente, está a nivel BAJO. Suponer que $\overline{PRE}$ y $\overline{CLR}$ permanecen a nivel ALTO.

FIGURA 8.86



18. Resolver el problema 17 con las mismas entradas J y K, pero con las entradas $\overline{PRE}$ y $\overline{CLR}$ que se muestran en la Figura 8.87 en función del reloj.

FIGURA 8.87

SECCIÓN 8.3 Flip-flops maestro-esclavo

19. Determinar la salida Q para las señales de la Figura 8.88. Suponer que Q es inicialmente un nivel BAJO.

20. Dibujar la salida Q del flip-flop B de la Figura 8.89 en relación adecuada al reloj. Los flip-flops se encuentran inicialmente en estado RESET.

FIGURA 8.88

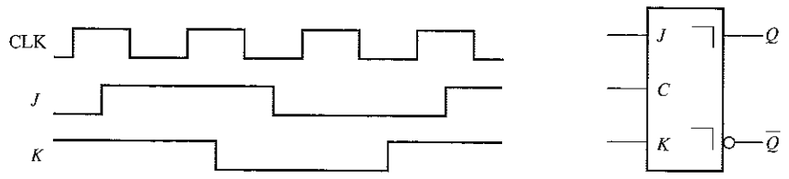
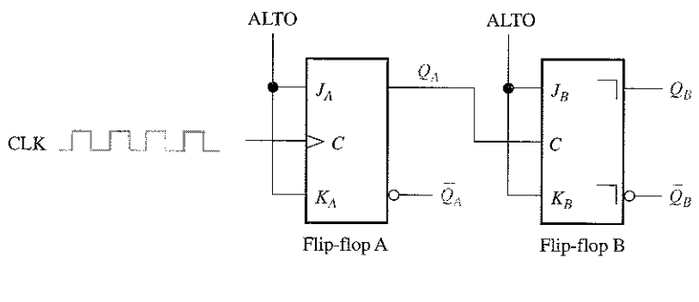
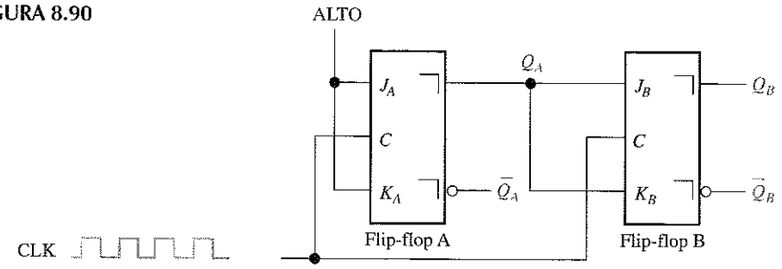


FIGURA 8.89



21. Dibujar la salida Q del flip-flop B de la Figura 8.90 en función de la señal de reloj. Los flip-flops se encuentran inicialmente en estado RESET.

FIGURA 8.90



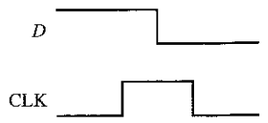
22. Para un flip-flop J-K maestro-esclavo con entrada de datos aislada (data lock-out), cuyas entradas son las que se muestran en la Figura 8.84, dibujar el diagrama de tiempos de Q, cuyo estado inicial es un nivel BAJO. Suponer que el reloj es activo a nivel ALTO.

23. La entrada D y un único impulso de reloj se muestran en la Figura 8.91. Comparar las salidas Q resultantes para flip-flops con disparo por flanco positivo, disparo por flanco negativo y maestro-esclavo de disparo por pulso. Los flip-flops se encuentran inicialmente en estado RESET.

SECCIÓN 8.4 Características de operación de los flip-flops

24. Típicamente, las hojas de especificaciones de los fabricantes especifican cuatro tipos de retardos asociados con los flip-flops. Nombrar y describir cada uno.

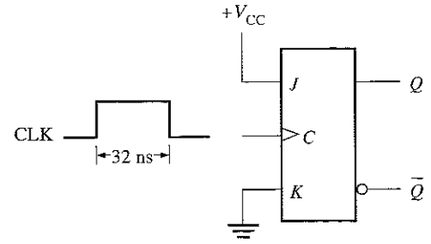
FIGURA 8.91



25. La hoja de especificaciones de un determinado flip-flop especifica que la duración mínima de un nivel ALTO para cada impulso de reloj es 30 ns y que la duración mínima para un nivel BAJO es de 37 ns. ¿Cuál es la frecuencia máxima de funcionamiento?

26. El flip-flop de la Figura 8.92 se encuentra inicialmente en RESET. Mostrar la relación entre la salida Q y el impulso de reloj, si el retardo de propagación t_{PLH} (del reloj a Q) es de 8 ns.

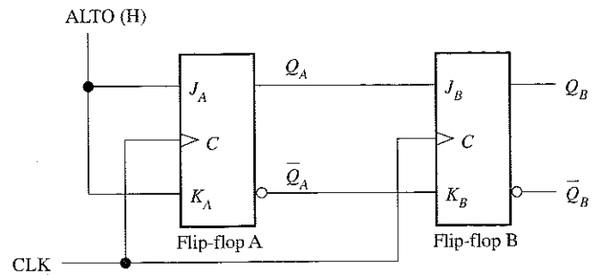
FIGURA 8.92



27. La corriente directa requerida por un determinado flip-flop que funciona a +5 V de tensión resulta ser de 10 mA. Un determinado dispositivo digital utiliza 15 de estos flip-flops. Determinar la capacidad de corriente requerida para la fuente de continua de +5 V y la potencia total disipada por el sistema.

28. Para el circuito de la Figura 8.93, determinar la frecuencia máxima de la señal del reloj para un funcionamiento fiable, si el tiempo de setup (establecimiento) de cada flip-flop es de 2 ns y los retardos de propagación (t_{PLH} y t_{PHL}) del reloj a la salida son de 5 ns para cada flip-flop.

FIGURA 8.93



SECCIÓN 8.5 Aplicaciones básicas de los flip-flops

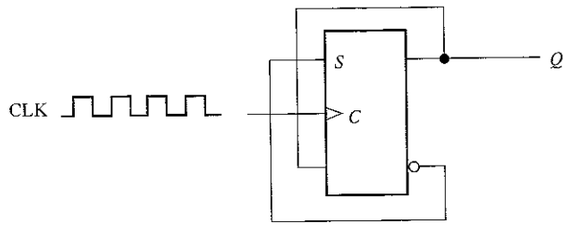
29. Un flip-flop D se encuentra conectado como se muestra en la Figura 8.94. Determinar la salida Q en función del reloj. ¿Cuál es la función que realiza este dispositivo?

30. Para el circuito de la Figura 8.93, desarrollar un diagrama de tiempos para ocho impulsos de reloj, mostrando las salidas Q_A y Q_B en función del reloj.

SECCIÓN 8.6 Monoestables

31. Determinar la anchura del impulso de un monoestable 74121, si la resistencia externa es de 3,3 kΩ y el condensador externo vale 2.000 pF.

FIGURA 8.94



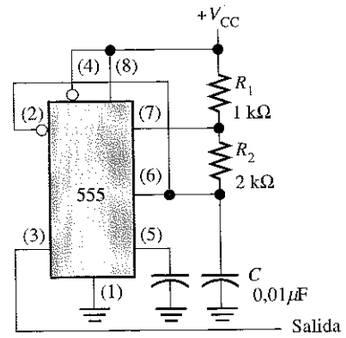
32. Se quiere generar un impulso de salida de 5 μs de duración con un monoestable 74LS122. Utilizando un condensador de 10.000 pF, determinar el valor de la resistencia externa requerida.

SECCIÓN 8.7 El temporizador 555

33. Diseñar un monoestable utilizando un temporizador 555 para producir un impulso de salida de 0,25 segundos.

34. Se configura un temporizador 555 para funcionar como multivibrador aestable, como se muestra en la Figura 8.95. Determinar su frecuencia.

FIGURA 8.95



35. Determinar los valores de las resistencias externas de un temporizador 555 utilizado como multivibrador aestable con frecuencia de salida de 20 kHz, si el condensador C vale 0,002 μF y el ciclo de trabajo es del 75 % aproximadamente.

SECCIÓN 8.8 Localización de averías

36. Se prueba el flip-flop de la Figura 8.96 bajo todas las posibles condiciones de entrada, tal como se muestra. ¿Está funcionando correctamente? Si no es así ¿cuál es la causa de fallo más probable?

37. Se utiliza una cuádruple puerta NAND 74HC00 para construir un latch S-R con entrada de habilitación en un prototipo de tarjeta de laboratorio, como muestra la Figura 8.97. El esquema de la parte (a) se utiliza para conectar el circuito de la parte (b). Cuando intentamos poner en funcionamiento el latch, nos encontramos con que la salida Q permanece a nivel ALTO independientemente de los valores de las entradas. Determinar cuál es el problema.

38. Determinar si el flip-flop de la Figura 8.98 está funcionando adecuadamente y, en caso contrario, identificar el fallo más probable.

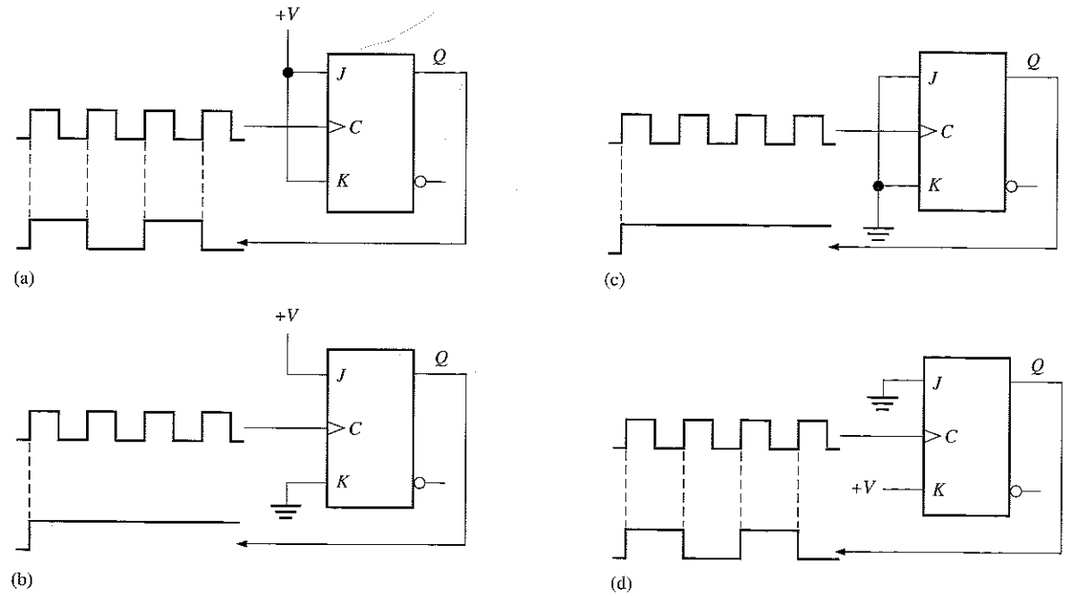


FIGURA 8.96

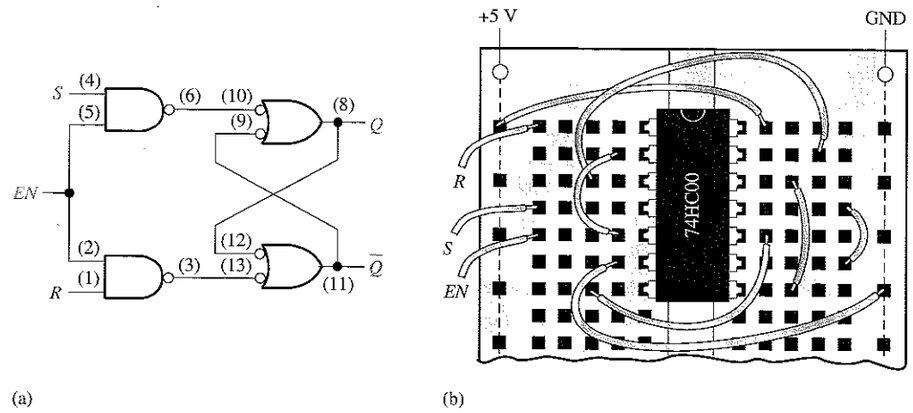
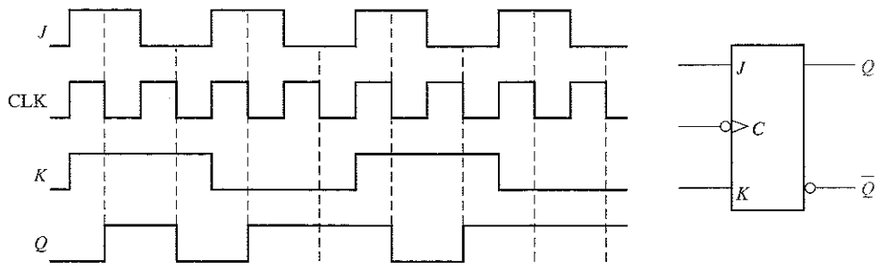


FIGURA 8.97

39. El circuito paralelo de almacenamiento de datos de la Figura 8.39 no funciona adecuadamente. Para depurarlo, primero nos aseguramos de que V_{CC} y tierra se encuentran conectados y, luego, aplicamos niveles bajos a todas las entradas D e introducimos impulsos en la línea del reloj. Se comprueba que las salidas Q están todas a nivel BAJO, por lo que por el momento todo es correcto. A continuación se aplican niveles altos a todas las entradas D y de nuevo se introducen impulsos en la línea del reloj. Cuando comprobamos las salidas Q, todavía permanecen a nivel BAJO. ¿Cuál es el problema y cuál sería el procedimiento que utilizaríamos para aislar el fallo a un único dispositivo?

FIGURA 8.98



40. El circuito del flip-flop de la Figura 8.99(a) se utiliza para generar una secuencia de cuenta binaria. Las puertas forman un decodificador que se supone que produce un nivel ALTO cuando ocurre un cero binario o el estado tres binario (00 o 11). Analizando las salidas Q_A y Q_B, se obtiene la imagen mostrada en la parte (b), que revela glitches en la salida del decodificador (X), además de los impulsos correctos. ¿Qué es lo que causa estos glitches y cómo se pueden eliminar?

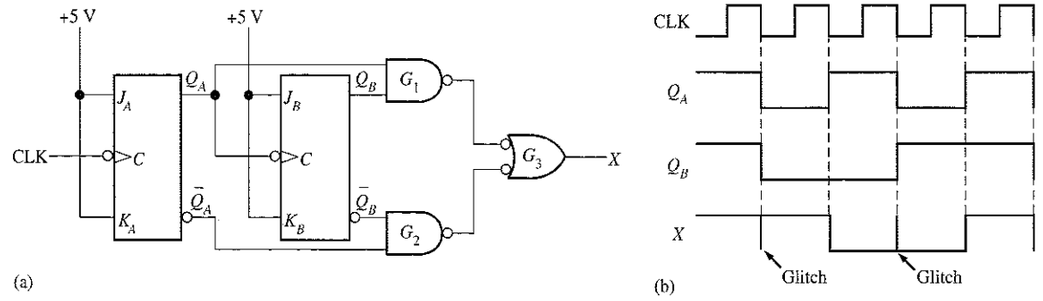


FIGURA 8.99

41. Determinar las salidas Q_A, Q_B y X durante seis impulsos de reloj en la Figura 8.99(a) para cada uno de los siguientes fallos en circuitos TTL. Inicialmente Q_A y Q_B están a nivel BAJO.

(a) La entrada J_A está en circuito abierto.

(b) La entrada K_B está en circuito abierto.

(c) La entrada Q_B está en circuito abierto.

(d) La entrada de reloj en el flip-flop B está cortocircuitada.

(e) La puerta G_2 está en circuito abierto.

42. Se conectan dos monoestables 74121 en una tarjeta, como se muestra en la Figura 8.100. Tras observar la pantalla del osciloscopio, ¿sacaríamos la conclusión de que el circuito está funcionando adecuadamente? En caso contrario, ¿cuál es la causa más probable del fallo?

SECCIÓN 8.9 Aplicación a los sistemas digitales

43. Utilizar temporizadores 555 para implementar los monoestables de 4 y 25 segundos para los circuitos de temporización del sistema. La entrada de disparo del 555 no puede permanecer a nivel BAJO después de una transición negativa, de forma que tenemos que desarrollar un circuito para producir impulsos negativos muy cortos, con el fin de disparar los temporizadores corto y largo cuando el sistema pasa por cada estado.

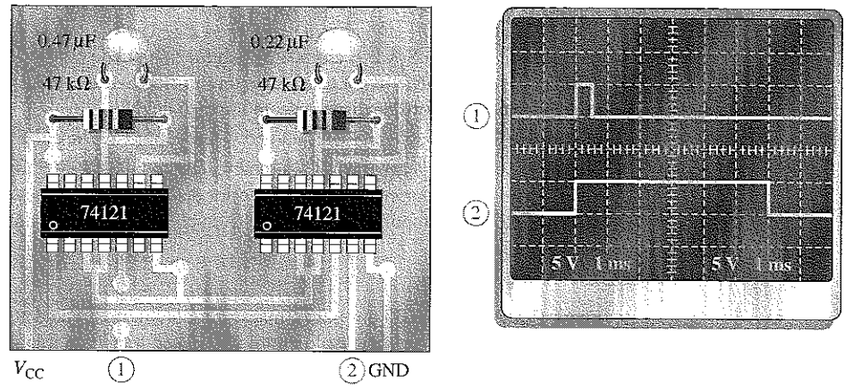


FIGURA 8.100

44. Modificar el circuito oscilador de 10 kHz de los circuitos de temporización del sistema de control de semáforos, para producir una salida con un ciclo de trabajo del 25%.

Problemas especiales de diseño

45. Diseñar un circuito contador básico que genere una secuencia binaria de cero a siete, utilizando flip-flops J-K disparados por flanco negativo.

46. En el departamento de logística de una fábrica de pelotas, éstas ruedan por una cinta y por una rampa hasta llegar a una caja. Cada pelota que pasa por la rampa activa un conmutador que produce un impulso eléctrico. La capacidad de cada caja es de 32 pelotas. Diseñar un circuito lógico para indicar cuándo una caja está llena, de forma que pueda ser sustituida por otra vacía.

47. Enumerar los cambios que serían necesarios en el sistema de control de semáforos para añadir una indicación de giro a la derecha de 15 segundos de duración. La indicación aparecería después de la luz roja y antes de la verde. Modificar el diagrama de estados del capítulo 6 (Figura 6.67) de manera que refleje estos cambios.

Localización de fallos con el banco de trabajo electrónico

48. Abrir el archivo PR08_48.EWB y probar los latches para determinar cuál falla.

49. Abrir el archivo PR08_49.EWB y probar los flip-flops J-K para determinar cuál falla.

50. Abrir el archivo PR08_50.EWB y probar los flip-flops D para determinar cuál falla.

51. Abrir el archivo PR08_51.EWB y probar los monoestables para determinar cuál falla.

52. Abrir el archivo PR08_52.EWB y probar el circuito divisor por cuatro, para determinar si falla. En caso afirmativo, identificar el fallo si es posible.

RESPUESTAS A LAS REVISIONES DE CADA SECCIÓN

SECCIÓN 8.1

1. Tres tipos de latches son el S-R, el S-R con entrada de habilitación y el D con entrada de habilitación.
2. $SR = 00$, NC; $SR = 01$, $Q = 0$; $SR = 10$, $Q = 1$; $SR = 11$, no válido
3. $Q = 1$

SECCIÓN 8.2

1. La salida de un latch S-R con entrada de habilitación puede cambiar siempre que la entrada de habilitación (*EN*) esté activa. La salida de un flip-flop S-R disparado por flanco puede cambiar sólo durante los flancos de disparo de un impulso de reloj.
2. El flip-flop J-K no tiene ningún estado no válido, como ocurre con el flip-flop S-R.
3. La salida Q se pone a nivel ALTO durante el flanco posterior del primer impulso del reloj, se pone a nivel BAJO durante el flanco posterior del segundo impulso, a nivel ALTO en el flanco posterior del tercer impulso y a nivel BAJO en el flanco posterior del cuarto impulso.

SECCIÓN 8.3

1. En un flip-flop disparado por pulso (maestro-esclavo), un bit de datos pasa al maestro durante el flanco anterior de un impulso de reloj y luego se transfiere a la salida (esclavo) durante el flanco posterior. En un flip-flop disparado por flanco, un bit de datos pasa a través del flip-flop y aparece en la salida durante el mismo flanco del reloj.
2. **(a)** No ocurre nada en el flip-flop disparado por flanco positivo.
   (b) La salida del flip-flop maestro-esclavo disparado por pulso cambiará.

SECCIÓN 8.4

1. **(a)** El tiempo de setup (establecimiento) es el tiempo que los datos de entrada deben estar presentes antes del flanco de disparo del impulso de reloj.
   (b) Tiempo de hold (mantenimiento) es el tiempo que los datos deben permanecer en la entrada después del flanco de disparo del impulso de reloj.
2. El 74AHC74 puede funcionar a la frecuencia máxima, de acuerdo con la Tabla 8.6.

SECCIÓN 8.5

1. Un registro es un grupo de flip-flops de almacenamiento de datos.
2. Para funcionar como divisor por dos, el flip-flop tiene que estar en modo de basculación ($J = 1$, $K = 1$).
3. Se necesitan seis flip-flops para formar un divisor por 64.

SECCIÓN 8.6

1. Un monoestable no redisparable no puede responder a otra entrada de disparo mientras que se encuentra en su estado inestable. Un monoestable disparable responde a cada entrada de disparo.
2. La anchura de los impulsos se ajusta mediante componentes R y C externos.

SECCIÓN 8.7

1. Un aestable no tiene estados estables. Un monoestable tiene un estado estable.
2. Ciclo de trabajo = (15 ms/20 ms) 100% = 75%

SECCIÓN 8.8

1. Sí, se puede utilizar un flip-flop D disparado por flanco negativo.
2. Se puede utilizar un temporizador 555 funcionando en modo de multivibrador aestable como reloj.

SECCIÓN 8.9

1. Los conmutadores se utilizan para simular la entrada de 2 bits en código Gray.
2. Los intervalos de 4s y de 25s ocurren en distintos estados.
3. De ninguna.

■ RESPUESTAS A LOS PROBLEMAS RELACIONADOS DE LOS EJEMPLOS

8.1 La salida Q es la misma que la mostrada en la Figura 8.5(b).
8.2 Véase la Figura 8.101.
8.3 Véase la Figura 8.102.
8.4 Véase la Figura 8.103.
8.5 Véase la Figura 8.104.
8.6 Véase la Figura 8.105.
8.7 Véase la Figura 8.106.
8.8 Véase la Figura 8.107.
8.9 Véase la Figura 8.108.

FIGURA 8.101

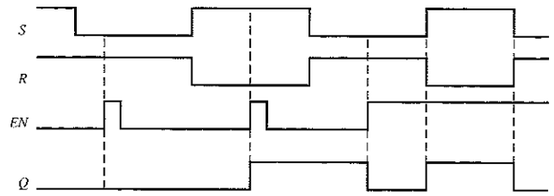


FIGURA 8.102

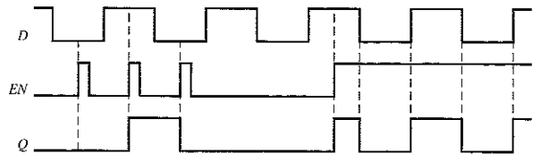


FIGURA 8.103

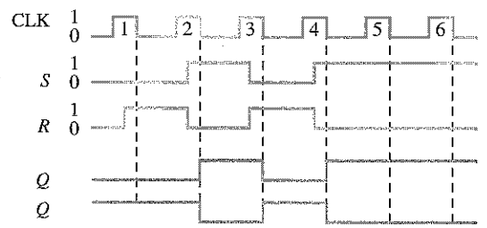


FIGURA 8.104

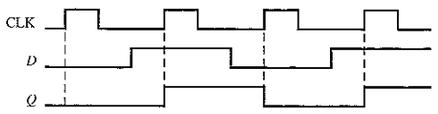


FIGURA 8.105

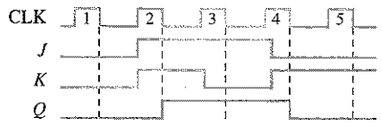


FIGURA 8.106

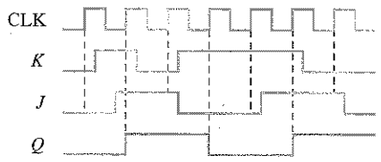


FIGURA 8.107

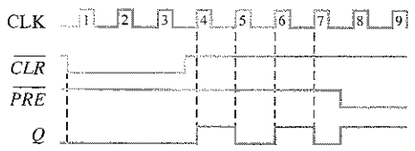
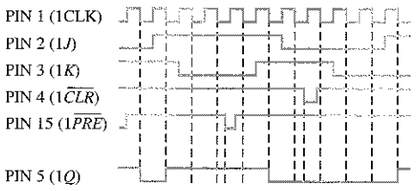


FIGURA 8.108



8.10 Véase la Figura 8.109.

8.11 $2^5 = 32$. Se requieren 5 flip-flops.

8.12 Dieciséis estados requieren cuatro flip-flops ($2^4 = 16$).

8.13 C_{EXT} = 7143 pF (6800 pF || 330 pF) conectado desde *CX* a *RX/CX* del 74121.

8.14 C_{EXT} = 560 pF, R_{EXT} = 27 kΩ. Véase la Figura 8.110.

8.15 R_{EXT} = 91 kΩ

8.16 Ciclo de trabajo ≅ 32%

FIGURA 8.109

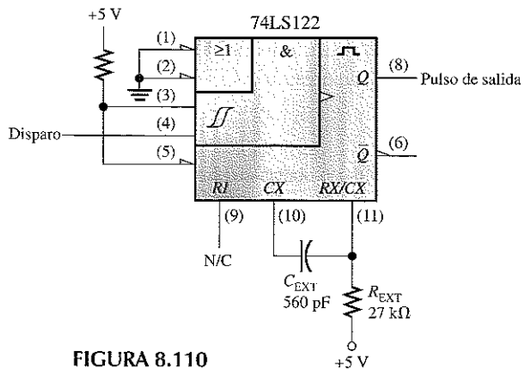


FIGURA 8.110

■ RESPUESTAS AL AUTOTEST

1. (a) **2.** (c) **3.** (d) **4.** (b) **5.** (d) **6.** (d) **7.** (a) **8.** (b)
9. (d) **10.** (d) **11.** (c) **12.** (f)

9

CONTADORES

9.1 Funcionamiento del contador asíncrono
9.2 Funcionamiento del contador síncrono
9.3 Contador síncrono ascendente/descendente
9.4 Diseño de contadores síncronos
9.5 Contadores en cascada
9.6 Decodificación de contadores
9.7 Aplicaciones de los contadores
9.8 Localización de averías
9.9 Símbolos lógicos con notación de dependencia
9.10 Aplicación a los sistemas digitales

WWW Simulación por computadora de circuitos digitales:
El tutorial de Electronics Workbench (EWB) se encuentra en la dirección http://www.prenhall.com/floyd.

■ OBJETIVOS DEL CAPÍTULO

- ❑ Describir la diferencia entre un contador asíncrono y un contador síncrono.
- ❑ Analizar los diagramas de tiempos de contadores.
- ❑ Analizar los circuitos contadores.
- ❑ Explicar en qué afecta el retardo de propagación al funcionamiento de un contador.
- ❑ Determinar el módulo de un contador.
- ❑ Modificar el módulo de un contador.
- ❑ Establecer las diferencias entre contadores binarios de 4 bits y contadores de décadas.
- ❑ Utilizar un contador ascendente/descendente para generar secuencias binarias directas e inversas.
- ❑ Determinar la secuencia de un contador.
- ❑ Utilizar circuitos integrados contadores en diversas aplicaciones.
- ❑ Diseñar un contador que pueda tener cualquier secuencia de estados especificada.
- ❑ Conectar varios contadores en cascada para conseguir módulos mayores.
- ❑ Utilizar puertas lógicas para decodificar cualquier estado de un contador.
- ❑ Eliminar los glitches en la decodificación de contadores.
- ❑ Explicar cómo funciona un reloj digital.
- ❑ Localizar averías en los contadores y resolver los distintos tipos de fallos.
- ❑ Interpretar los símbolos lógicos de los contadores que usan notación de dependencia.
- ❑ Utilizar un contador en una aplicación de sistemas.

■ INTRODUCCIÓN

Como ya se ha visto en el Capítulo 8, los flip-flops pueden conectarse entre sí para realizar funciones de recuento. A esta combinación de flip-flops se la denomina contador. El número de flip-flops que se utilizan y la forma en que se conectan determinan el número de estados (que recibe el nombre de módulo) y también la secuencia específica de estados por los que pasa el contador durante un ciclo completo.

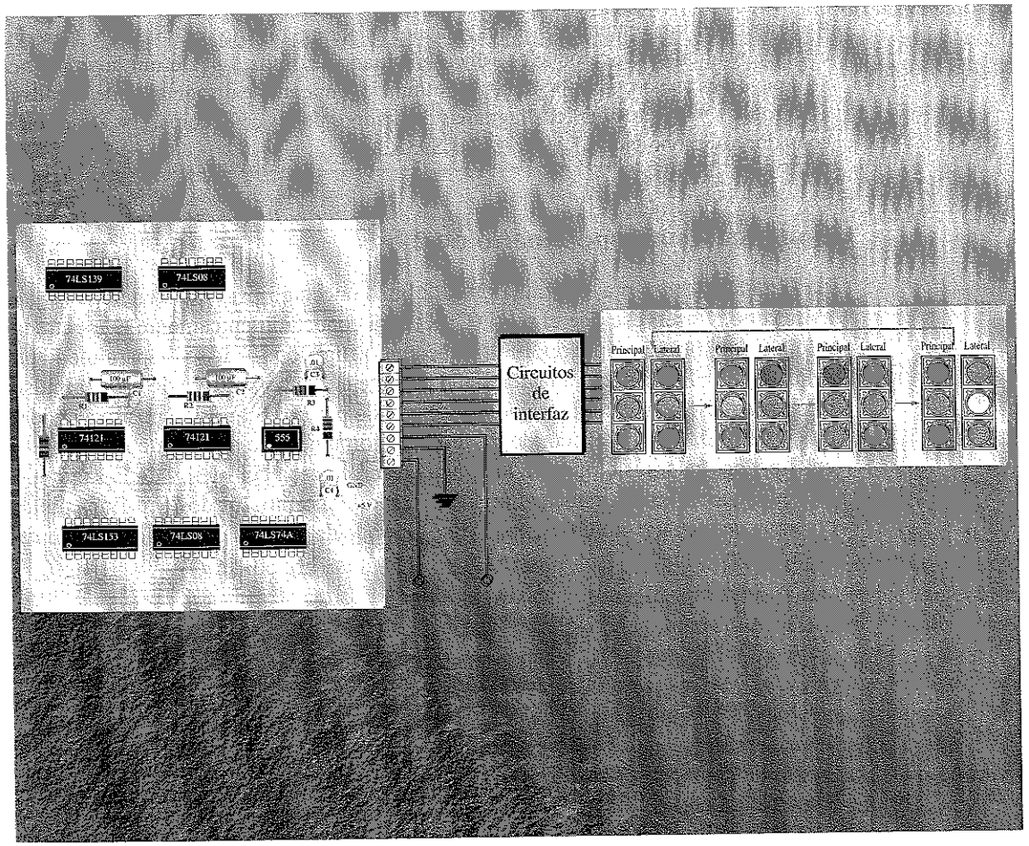
Dependiendo del modo en que se aplique la señal de reloj, los contadores se clasifican en dos amplias categorías: asíncronos y síncronos. En los contadores asíncronos, normalmente denominados *contadores con propagación (ripple counters)*, se aplica una señal de reloj externa a la entrada de reloj del primer flip-flop y luego a los siguientes flip-flops se les aplica la señal de reloj mediante la salida del flip-flop anterior. En los contadores síncronos, la entrada de reloj se conecta a todos los flip-flops, de forma que se les aplica la señal de reloj simultáneamente. Dentro de cada una de estas dos categorías, los contadores se clasifican por el tipo de secuencia, el número de estados o el número de flip-flops del contador.

DISPOSITIVOS ESPECÍFICOS

74LS93A	74HC160	74HC161
74HC163	74LS190	74LS247

■ APLICACIÓN A LOS SISTEMAS DIGITALES

Esta aplicación a los sistemas digitales ilustra los conceptos que se tratan en el capítulo. La aplicación de la Sección 9.10 continúa con el sistema de control de semáforos de los últimos tres capítulos. Este capítulo se ocupa de la lógica secuencial del sistema que produce la secuencia de luces basada en las entradas de los circuitos de temporización y del sensor de vehículos.



9.1 ■ FUNCIONAMIENTO DEL CONTADOR ASÍNCRONO

El término asíncrono se refiere a los sucesos que no poseen una relación temporal fija entre ellos y que, generalmente, no ocurren al mismo tiempo. Un contador asíncrono es aquel en el que los flip-flops (FF) del contador no cambian de estado exactamente al mismo tiempo, dado que no comparten el mismo impulso de reloj.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir el funcionamiento de un contador asíncrono binario de 2 bits. ■ Describir el funcionamiento de un contador asíncrono binario de 3 bits. ■ Definir la *propagación* en contadores asíncronos. ■ Describir el funcionamiento de un contador de décadas asíncrono. ■ Desarrollar los diagramas de tiempos de los contadores. ■ Describir el contador asíncrono binario de 4 bits 74LS93A.

Contador asíncrono binario de 2 bits

La Figura 9.1 presenta un contador de 2 bits conectado para que funcione en modo **asíncrono**. Obsérvese que el reloj (CLK) está conectado únicamente a la entrada de reloj (C) del primer flip-flop, FF0. El segundo flip-flop, FF1, se dispara mediante la salida $\overline{Q}_0$ de FF0. FF0 cambia de estado durante el flanco positivo de cada impulso de reloj, pero FF1 sólo cambia cuando es disparado por una transición positiva de la salida $\overline{Q}_0$ de FF0. Debido al retardo de propagación inherente al paso de las señales por un flip-flop, las transiciones de los impulsos de entrada del reloj y de la salida $\overline{Q}_0$ de FF0 no pueden ocurrir nunca al mismo tiempo. Por tanto, los dos flip-flops nunca se disparan de forma simultánea, por lo que el modo de funcionamiento de este contador es asíncrono.

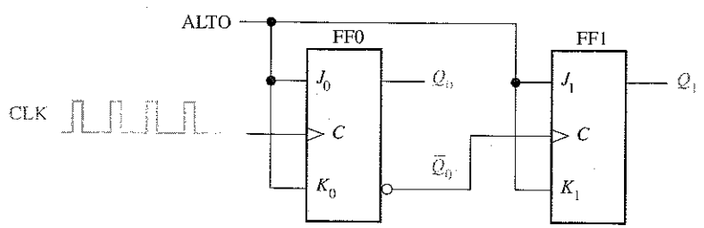


FIGURA 9.1
Contador asíncrono binario de 2 bits.

Diagrama de tiempos. Vamos a examinar el funcionamiento básico del **contador asíncrono** de la Figura 9.1, aplicando cuatro impulsos de reloj a FF0 y observando la salida Q de cada flip-flop. La Figura 9.2 ilustra los cambios de estado en las salidas del flip-flop en respuesta a los impulsos de reloj. Ambos flip-flops están conectados en modo de basculación (J =1, K =1) y se presupone que, inicialmente, están en estado RESET (Q a nivel BAJO).

El flanco positivo de CLK1 (impulso de reloj 1) hace que la salida Q_0 de FF0 pase a nivel ALTO, como muestra la Figura 9.2. Al mismo tiempo, la salida $\overline{Q}_0$ pasa a

nivel BAJO, pero esto no afecta a FF1, ya que tiene que ser una transición positiva la que le dispare. Después del flanco anterior de CLK1, $Q_0 = 1$ y $Q_1 = 0$. El flanco positivo de CLK2 hace que Q_0 pase a nivel BAJO. La salida $\overline{Q}_0$ se pone a nivel ALTO y dispara FF1, haciendo que Q_1 pase a nivel ALTO. Tras el flanco anterior de CLK2, $Q_0 = 0$ y $Q_1 = 1$. El flanco positivo de CLK3 hace que Q_0 pase a nivel ALTO de nuevo. La salida $\overline{Q}_0$ se pone a nivel BAJO y no afecta al estado de FF1. Por tanto, tras el flanco anterior de CLK3, $Q_0 = 1$ y $Q_1 = 1$. El flanco positivo de CLK4 hace que Q_0 pase a nivel BAJO, mientras que $\overline{Q}_0$ se pone a nivel ALTO y dispara FF1, haciendo que Q_1 pase a nivel BAJO. Después del flanco anterior de CLK4, $Q_0 = 0$ y $Q_1 = 0$. El contador ha vuelto a su estado original (los dos flip-flops se encuentran en estado RESET).

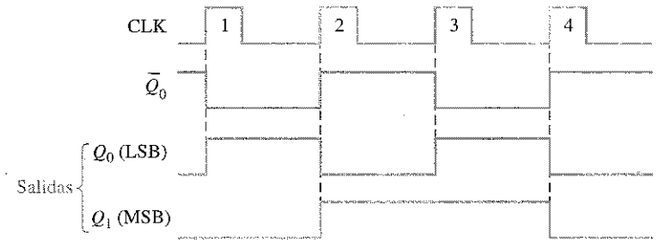


FIGURA 9.2
Diagrama de tiempos del contador de la Figura 9.1.

En el cronograma, las formas de onda de las salidas Q_0 y Q_1 se muestran en función de los impulsos de reloj, como ilustra la Figura 9.2. Para simplificar, las transiciones de Q_0, Q_1 y los impulsos de reloj se muestran como simultáneos, aunque se trate de un contador asíncrono. Existe, por supuesto, un ligero retardo entre las transiciones de CLK y Q_0 y las transiciones de $\overline{Q}_0$ y $Q1$.

Obsérvese en la Figura 9.2, que el contador de 2 bits dispone de cuatro estados diferentes, como cabría esperar de dos flip-flops ($2^2 = 4$). Además, téngase en cuenta que si Q_0 representa el bit menos significativo (LSB) y Q_1 representa el bit más significativo (MSB), la secuencia de los estados del contador representa una secuencia de números binarios, como se muestra en la Tabla 9.1. *En la lógica digital, Q_0 es siempre el bit menos significativo (LSB) a menos que se indique lo contrario.*

TABLA 9.1
Secuencia de estados binarios para el contador de la Figura 9.1

Pulso de reloj	Q_1	Q_0
Inicialmente	0	0
1	0	1
2	1	0
3	1	1
4 (nuevo ciclo)	0	0

Puesto que pasa por una **secuencia** binaria, el contador de la Figura 9.1 es un contador binario. En realidad, cuenta el número de impulsos de reloj hasta el tercero

y, en el cuarto impulso, inicia un nuevo ciclo a partir de su estado original ($Q_0 = 0$, $Q_1 = 0$). El inicio de un nuevo ciclo (**recycle**, término que se aplica comúnmente al funcionamiento de los contadores) se refiere a la transición del contador de su *estado final* a su *estado original*.

Contador asíncrono binario de 3 bits

En la Figura 9.3(a) se muestra un contador asíncrono binario de 3 bits. Su funcionamiento básico es el mismo que el del contador de 2 bits que se acaba de estudiar, excepto en que el contador de 3 bits tiene ocho estados, debido a que está formado por tres flip-flops. En la Figura 9.3(b) se presenta un diagrama de tiempos para ocho impulsos de reloj. Obsérvese que el contador de la Figura 9.3 avanza a través de una secuencia binaria desde cero hasta siete, iniciando después un nuevo ciclo desde su estado cero. Esta secuencia del contador se muestra en la Tabla 9.2.

TABLA 9.2
Secuencia de estados de un contador binario de tres bits.

Pulso de reloj	Q_2	Q_1	Q_0
Inicialmente	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8 (nuevo ciclo)	0	0	0

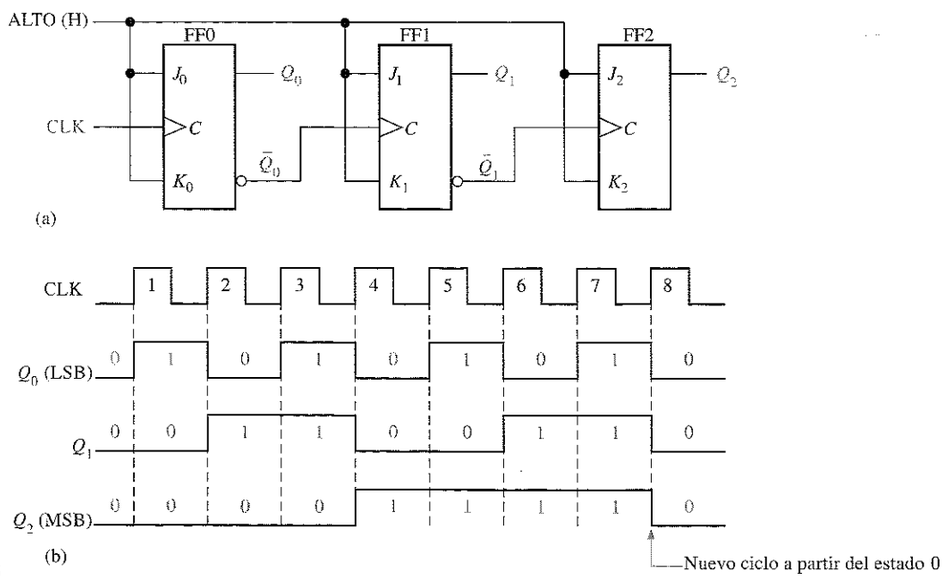
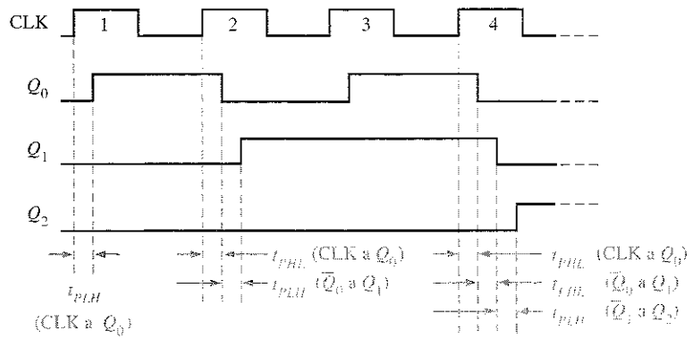


FIGURA 9.3
Contador binario asíncrono de tres bits y su diagrama de tiempos para un ciclo.

Retardo de propagación. A los contadores asíncronos también se les denomina **contadores con propagación** por la siguiente razón: el efecto de un impulso en la entrada de reloj lo "siente" primero FF0. Este efecto no llega inmediatamente a FF1, debido al retardo de propagación a través de FF0. Del mismo modo, se produce un retardo de propagación a través de FF1, antes de que FF2 pueda ser disparado. Por tanto, el efecto de un impulso en la entrada de reloj se "propaga" a través del contador, tardando un cierto tiempo en alcanzar el último de los flip-flops, debido a los retardos de propagación.

Como ilustración, obsérvese que todos los flip-flops del contador de la Figura 9.3 cambian de estado en el flanco anterior de CLK4. Este efecto de propagación de la señal de reloj se muestra en la Figura 9.4 para los cuatro primeros impulsos de reloj, indicando los retardos de propagación. La transición de nivel ALTO a nivel BAJO de Q_0 se produce después de un determinado retardo (t_{PHL}) después de la transición positiva del impulso de reloj. La transición de nivel ALTO a nivel BAJO de Q_1 ocurre un tiempo (t_{PHL}) después de la transición positiva de $\bar{Q}_0$. La transición de nivel BAJO a nivel ALTO de Q_2 se produce después de otra unidad de retardo (t_{PHL}) después de la transición positiva de $\bar{Q}_1$. Como puede verse, FF2 no se dispara hastaque han transcurrido dos unidades de retardo después del flanco positivo del impulso de reloj, CLK4. Por tanto, se necesitan tres unidades de retardo para que el efecto del impulso de reloj CLK4 se propague a través del contador y Q_2 pase de nivel BAJO a nivel ALTO.

FIGURA 9.4
Retardos de propagación en un contador binario asíncrono (con propagación de reloj) de 3 bits.



Este retardo acumulativo de un contador asíncrono es una de sus mayores desventajas para muchas aplicaciones, ya que limita la velocidad a la que el contador puede ser sincronizado, y puede dar lugar a problemas de decodificación. El retardo acumulativo máximo en un contador tiene que ser menor que el periodo de la señal de reloj.

EJEMPLO 9.1 En la Figura 9.5(a) se muestra un contador asíncrono binario de 4 bits. Cada flip-flop es disparado por flanco negativo y tiene un retardo de propagación de 10 nanosegundos (ns). Dibujar un diagrama de tiempos que muestre la salida Q de cada uno de los flip-flops y determinar el retardo de propagación total desde el flanco de disparo de un impulso de reloj hasta que pueda producirse el cambio correspondiente en el estado de

Q_3. Determinar también la frecuencia máxima de reloj a la que puede funcionar el contador.

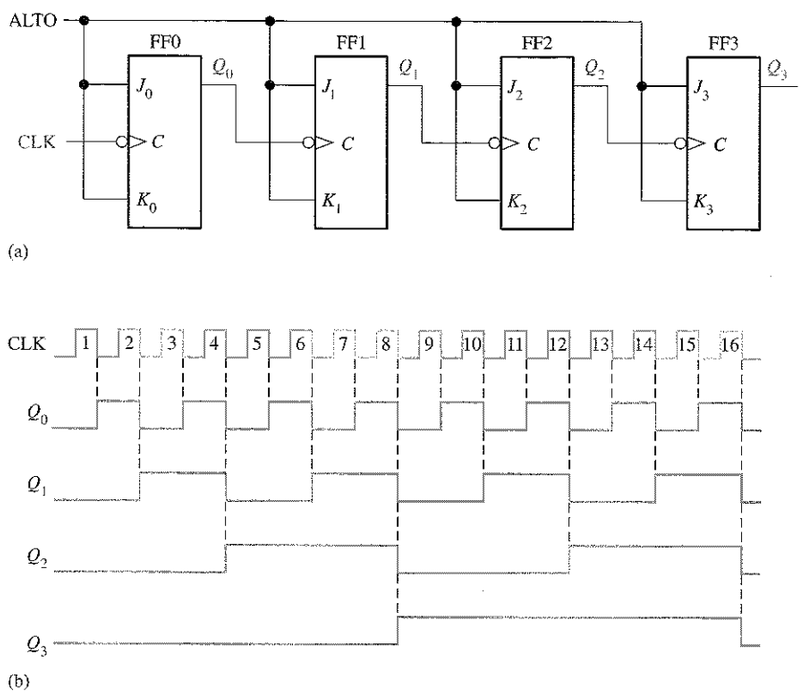


FIGURA 9.5
Contador binario asíncrono de 4 bits y su diagrama de tiempos.

Solución. En la Figura 9.5(b) se muestra el diagrama de tiempos, habiendo omitido los retardos. Por lo que se refiere al retardo total, el efecto de CLK8 o CLK16 se tiene que propagar a través de cuatro flip-flops antes de que Q_3 cambie, de forma que:

$$t_{p(tot)} = 4 \times 10 \text{ ns} = \mathbf{40 \text{ ns}}$$

La frecuencia máxima de reloj es:

$$f_{\max} = \frac{1}{t_{p(tot)}} = \frac{1}{40 \text{ ns}} = \mathbf{25 \text{ MHz}}$$

Problema relacionado. Dibujar el diagrama de tiempos si todos los flip-flops de la Figura 9.5(a) fueran disparados por flanco positivo.

Contador de décadas asíncrono

El **módulo** de un contador es el número de estados distintos por el que el contador puede pasar de forma secuencial. El número máximo de posibles estados (módulo máximo) de un contador es 2^n, donde n representa el número de flip-flops del contador.

También se pueden diseñar contadores que tengan un número de estados en su secuencia que sea menor que el máximo de 2^n. La secuencia resultante se denomina **secuencia truncada**.

Un módulo típico en los contadores con secuencia **truncada** es diez (denominado MOD10). Los contadores que tienen diez estados en su secuencia se denominan **contadores de décadas**. Un contador de **décadas**, cuya secuencia de cuenta vaya de cero (0000) a nueve (1001), es un contador de décadas BCD, ya que su secuencia de diez estados corresponde al código BCD. Este tipo de contadores resulta muy útil en las aplicaciones de displays, en las que se necesitan códigos BCD para la conversión a código decimal.

Para obtener una secuencia truncada, es necesario forzar al contador a que inicie un nuevo ciclo antes de haber pasado por todos los estados normales. Por ejemplo, el contador BCD de décadas tiene que comenzar de nuevo en el estado 0000 después de pasar por el estado 1001. Un contador de décadas requiere cuatro flip-flops (tres serían insuficientes, ya que $2^3 = 8$).

Para ilustrar el funcionamiento de los contadores truncados, vamos a utilizar un contador asíncrono de 4 bits, como el de la Figura 9.5(a), en el que modificaremos su secuencia. Una manera de hacer que un contador inicie un nuevo ciclo después de haber llegado a nueve (1001) consiste en decodificar el diez (1010) con una puerta NAND y conectar la salida de la puerta NAND a las entradas de borrado ($\overline{CLR}$) de los flip-flops, como se muestra en la Figura 9.6(a).

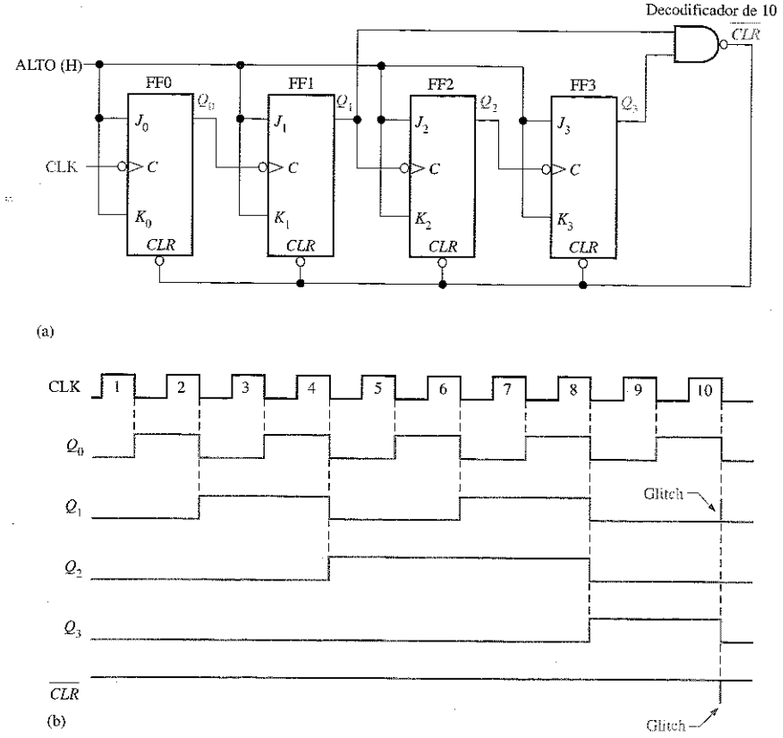


FIGURA 9.6
Contador de décadas asíncrono con reinicialización asíncrona.

Decodificación parcial. Obsérvese en la Figura 9.6(a) que únicamente Q_1 y Q_3 están conectadas a las entradas de la puerta NAND. Esta disposición es un ejemplo de *decodificación parcial*, mediante la cual dos únicos estados ($Q_1 = 1$ y $Q_3 = 1$) son suficientes para decodificar el valor diez, ya que ninguno de los otros estados (de cero a nueve) tienen Q_1 y Q_3 a nivel ALTO al mismo tiempo. Cuando el contador llega al número diez (1010), la salida de la puerta decodificadora pasa a nivel BAJO y pone a cero asíncronamente todos los flip-flops.

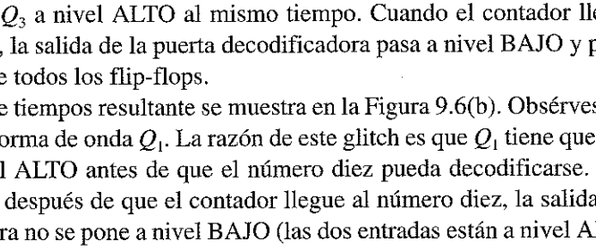
El diagrama de tiempos resultante se muestra en la Figura 9.6(b). Obsérvese que hay un glitch en la forma de onda Q_1. La razón de este glitch es que Q_1 tiene que pasar primero por el nivel ALTO antes de que el número diez pueda decodificarse. Hasta unos nanosegundos después de que el contador llegue al número diez, la salida de la puerta decodificadora no se pone a nivel BAJO (las dos entradas están a nivel ALTO). Por tanto, el contador se encuentra en el estado 1010 durante un corto periodo de tiempo antes de que se reinicialice a 0000, produciendo, por tanto, el glitch en Q_1 y en la línea $\overline{CLR}$ que sirve para poner a cero el contador.

Como muestra el Ejemplo 9.2, se pueden implementar otras secuencias truncadas de manera similar.

EJEMPLO 9.2 Explicar cómo se puede implementar un contador asíncrono que tenga módulo doce con una secuencia binaria directa desde 0000 hasta 1011.

Solución. Puesto que tres flip-flops pueden producir un máximo de ocho estados, necesitamos cuatro flip-flops para producir cualquier módulo mayor que ocho y menor o igual que dieciséis.

Cuando el contador alcanza el estado final 1011 tiene que iniciar un nuevo ciclo a partir de 0000, en lugar de pasar al siguiente estado natural, 1100, como ilustra la siguiente secuencia:



Obsérvese que, en cualquier caso, tanto Q_0 como Q_1 toman el valor 0, mientras que a Q_2 y Q_3 se les debe forzar para que tomen el valor 0 en el duodécimo impulso de reloj. La Figura 9.7(a) muestra el contador de módulo 12. La puerta NAND decodifica parcialmente el número doce (1100) y pone a cero los flip-flops 2 y 3. Por tanto, en el duodécimo impulso de reloj, se fuerza al contador a iniciar un nuevo ciclo, pasando de once a cero, como se muestra en el diagrama de tiempos de la Figura 9.7(b). Permanece en el número doce sólo durante unos cuantos nanosegundos antes de ponerse a cero por el glitch en $\overline{CLR}$.

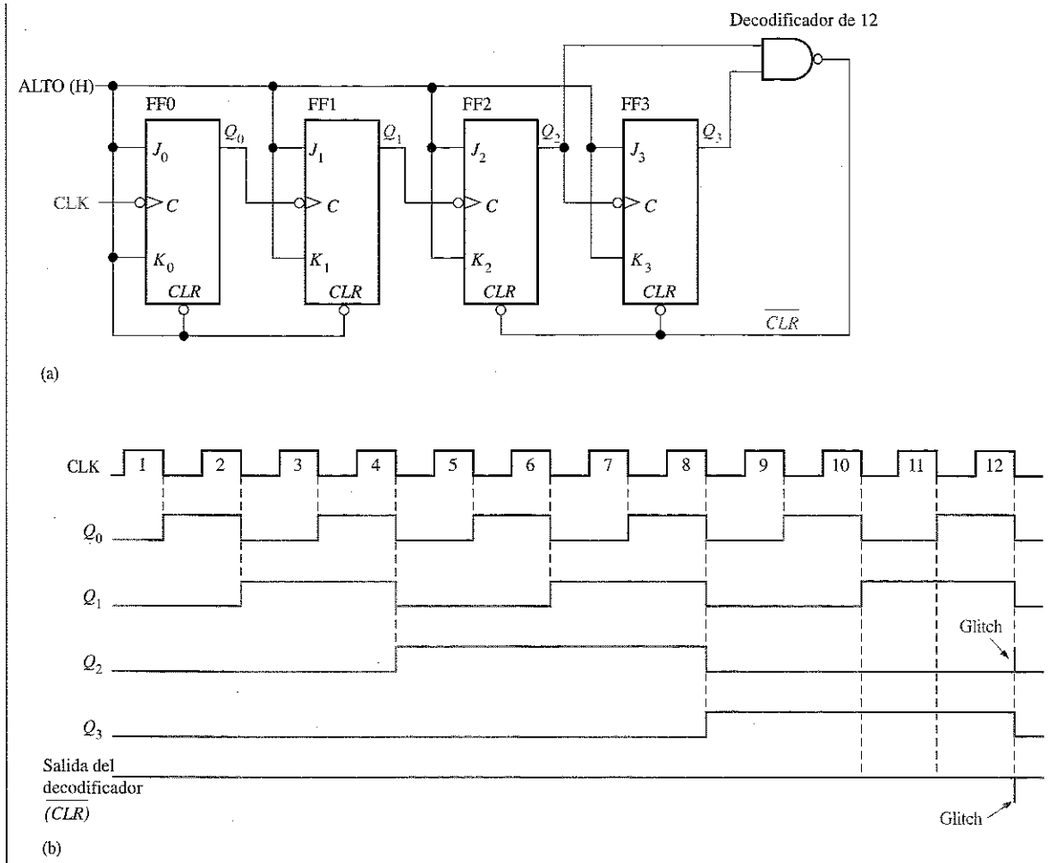


FIGURA 9.7
Contador de módulo 12 con temporización y reinicialización asíncronas.

Problema relacionado. ¿Cómo se puede modificar el contador de la Figura 9.7(a) para hacer de él un contador de módulo 13?

Un contador binario asíncrono de 4 bits

El 74LS93A es un ejemplo de circuito integrado contador asíncrono. Como muestra el diagrama lógico de la Figura 9.8, este dispositivo está formado por un flip-flop y un contador asíncrono de 3 bits. Esta disposición le proporciona una gran flexibilidad. Si se utiliza únicamente el flip-flop, se puede utilizar como dispositivo divisor por 2; y si se utiliza únicamente el contador de 3 bits, se puede emplear como contador de módulo 8. Este dispositivo proporciona además entradas de puesta a cero (RESET) *RO(1)* y *RO(2)*. Cuando estas dos entradas están a nivel ALTO, el contador se resetea al estado 0000 mediante $\overline{CLR}$.

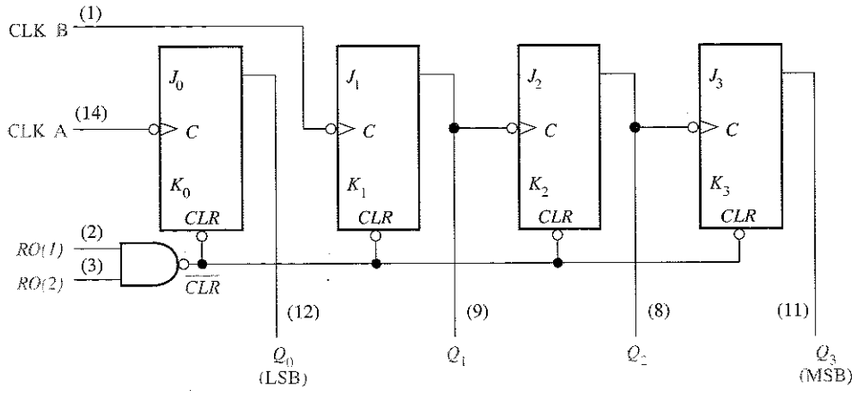
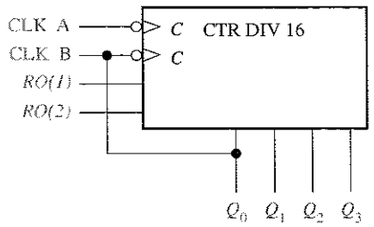
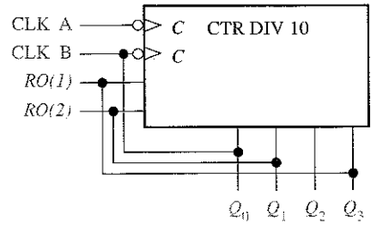


FIGURA 9.8
Diagrama lógico del contador binario asíncrono de 4 bits 74LS93A. Los números de los pines se indican entre paréntesis. Todas las entradas J y K están internamente conectadas a nivel ALTO.

Adicionalmente, el 74LS93A se puede utilizar como contador de 4 bits de módulo 16 (cuenta de cero a 15), conectando la salida Q_0 a la entrada CLK B, como muestra la Figura 9.9(a). También se puede configurar como contador de décadas (cuenta de 0 a 9) con reinicialización asíncrona, utilizando las entradas de puesta a cero para decodificar parcialmente el número diez, como muestra la Figura 9.9(b).



(a) 74LS93A conectado como contador de módulo 16



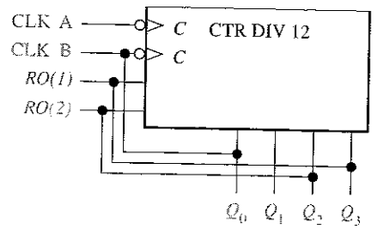
(b) 74LS93A conectado como contador de décadas

FIGURA 9.9
Dos configuraciones del contador asíncrono 74LS93A. CTR DIV n, indica un contador de n estados.

EJEMPLO 9.3 Explicar cómo se puede usar un 74LS93A como contador de módulo 12.

Solución. Utilizar las entradas de puesta a cero *RO(1)* y *RO(2)*, para decodificar parcialmente el número 12 (recuérdese que hay una puerta NAND interna asociada a estas entradas). La decodificación del número 12 se lleva cabo conectando Q_3 a *RO(1)* y Q_2 a *RO(2)*, como se muestra en la Figura 9.10. La salida Q_0 se conecta a CLK B para conseguir un contador de 4 bits.

FIGURA 9.10
74LS93A conectado como contador de módulo 12.



Inmediatamente después de que el contador alcanza el estado 12 (1100), vuelve al estado inicial 0000. El inicio de un nuevo ciclo, sin embargo, origina un glitch en Q_2 debido a que el contador tiene que permanecer en el estado 1100 durante unos pocos nanosegundos antes de comenzar otro ciclo.

Problema relacionado. Explicar cómo se podría conectar un 74LS93A como contador de módulo 13.

REVISIÓN DE LA SECCIÓN 9.1

1. ¿Qué significa el término *asíncrono* en relación a los contadores?
2. ¿Cuántos estados tiene un contador de módulo 14? ¿Cuál es el mínimo número de flip-flops requerido?

9.2 ■ FUNCIONAMIENTO DEL CONTADOR SÍNCRONO

El término síncrono se refiere a los eventos que tienen una relación temporal fija entre sí. Con respecto al funcionamiento del contador, síncrono significa que todos los flip-flops del contador reciben en el mismo instante la señal de reloj.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir el funcionamiento de un contador síncrono binario de 2 bits. ■ Describir el funcionamiento de un contador síncrono binario de 3 bits. ■ Describir el funcionamiento de un contador síncrono binario de 4 bits. ■ Describir el funcionamiento de un contador de décadas síncrono. ■ Desarrollar los diagramas de tiempos de los contadores. ■ Utilizar el contador binario de 4 bits 74HC163 y el contador BCD de décadas 74HC160.

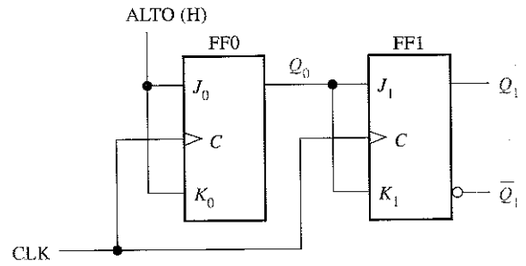
Contador binario síncrono de 2 bits

La Figura 9.11 muestra un contador binario **síncrono** de 2 bits. Obsérvese que debe utilizarse una disposición distinta a la del contador asíncrono para las entradas J_1 y K_1 de FF1, con el fin de poder conseguir una secuencia binaria.

El funcionamiento de este **contador síncrono** es el siguiente: en primer lugar, se supone que el contador se encuentra inicialmente en el estado binario 0; es decir, los dos flip-flops se encuentran en estado RESET. Cuando se aplica el flanco positivo del

primer impulso de reloj, FF0 bascula, por lo que Q_0 se pone a nivel ALTO. ¿Qué le ocurre a FF1 en el flanco positivo de CLK1? Para averiguarlo, vamos a fijarnos en las condiciones de entrada de FF1. Las entradas J_1 y K_1 están ambas a nivel BAJO, ya que están conectadas a Q_0, y ésta todavía no se ha puesto a nivel ALTO. Recuérdese que existe un retardo de propagación desde el flanco de disparo del impulso de reloj hasta que, realmente, se realiza la transición en la salida Q. Por tanto, $J = 0$ y $K = 0$ cuando se aplica el flanco anterior del primer impulso de reloj. Ésta es una condición de no cambio y, por tanto, FF1 no cambia de estado. En la Figura 9.12(a) se muestra una parte del diagrama de tiempos de esta fase del funcionamiento del contador.

FIGURA 9.11
Contador binario síncrono de dos bits.



Después de CLK1, $Q_0 = 1$ y $Q_1 = 0$ (que corresponde al estado binario 1). Cuando se produce el flanco anterior de CLK2, FF0 bascula y Q_0 se pone a nivel BAJO. Puesto que FF1 tiene un nivel ALTO ($Q_0 = 1$) en sus entradas J_1 y K_1 durante el flanco de disparo del impulso de reloj, el flip-flop bascula y Q_1 pasa a nivel ALTO. Por tanto, después de CLK2, $Q_0 = 0$ y $Q_1 = 1$ (que corresponde al estado binario 2). En la Figura 9.12(b) se muestra en detalle esta parte del diagrama de tiempos para esta condición.

Cuando se produce el flanco anterior de CLK3, FF0 bascula de nuevo al estado SET ($Q_0 = 1$) y FF1 permanece en estado SET ($Q_1 = 1$), ya que sus entradas J_1 y K_1 están ambas a nivel BAJO ($Q_0 = 0$). Tras este flanco de disparo, $Q_0 = 1$ y $Q_1 = 1$ (que corresponde al estado binario 3). En la Figura 9.12(c) se muestra en detalle el diagrama de tiempos para esta condición.

Finalmente, durante el flanco anterior de CLK4, Q_0 y Q_1 se ponen a nivel BAJO, dado que ambos flip-flops están en modo de basculación debido al valor presente en sus entradas J y K. En la Figura 9.12(d) se muestra en detalle el diagrama de tiempos para esta condición. El contador acaba de iniciar un nuevo ciclo a partir de su estado original, 0 binario.

El diagrama de tiempos completo del contador de la Figura 9.11 se muestra en la Figura 9.13. Obsérvese que todas las transiciones de las señales son coincidentes; es decir, no se indican los retardos de propagación. Aunque los retardos son un factor importante en el funcionamiento de un contador síncrono, se suelen omitir para simplificar los diagramas de tiempos generales. Si no se muestran los pequeños retardos y las diferencias de temporización, se puede conseguir relacionar mejor las señales resultantes de un circuito lógico. Sin embargo, en circuitos digitales de alta velocidad, estos pequeños retardos son una consideración importante en el diseño y la localización de averías.

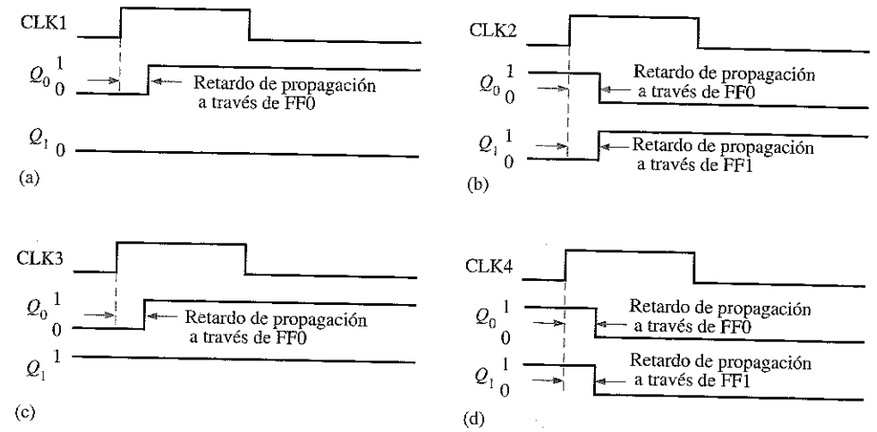
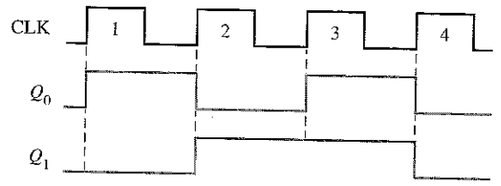


FIGURA 9.12
Diagramas de tiempos para un contador síncrono de 2 bits (los retardos de propagación de ambos flip-flops se consideran iguales).

FIGURA 9.13
Diagrama de tiempos del contador de la Figura 9.11.



Contador síncrono binario de 3 bits

En la Figura 9.14 se muestra un contador síncrono binario de 3 bits y en la Figura 9.15 su diagrama de tiempos. Para entender el funcionamiento de este tipo de contador debe examinarse detenidamente su secuencia de estados, la cual se muestra en la Tabla 9.3.

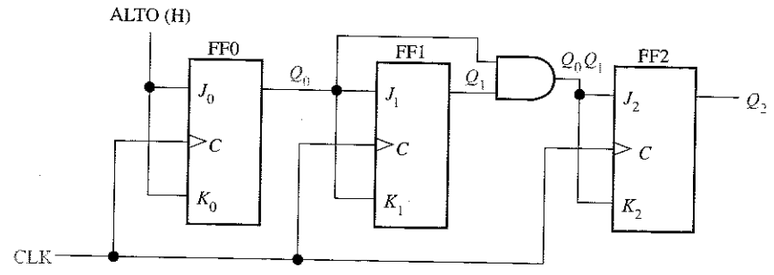


FIGURA 9.14
Contador binario síncrono de 3 bits.

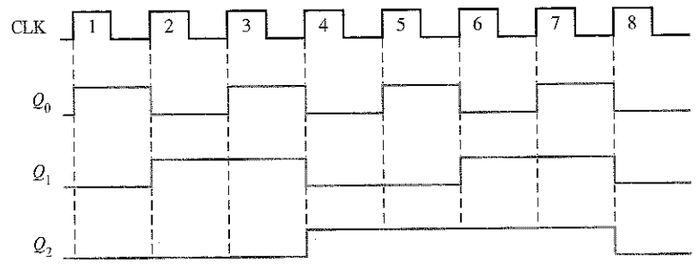


FIGURA 9.15
Diagrama de tiempos del contador de la Figura 9.14.

TABLA 9.3
Secuencia de estados del contador binario de tres bits.

Pulso de reloj	Q_2	Q_1	Q_0
Inicialización	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8 (nuevo ciclo)	0	0	0

En primer lugar, vamos a fijarnos en Q_0. Obsérvese que, Q_0 cambia en cada impulso de reloj a medida que el contador avanza desde su estado original hasta su estado final, para luego iniciar un nuevo ciclo a partir del estado original. Para conseguir este funcionamiento, FF0 tiene que mantenerse en modo de basculación, aplicando constantemente niveles altos en sus entradas J_0 y K_0. Téngase en cuenta que Q_1 pasa al estado contrario cada vez que Q_0 está a 1. Este cambio se produce en CLK2, CLK4, CLK6 y CLK8. El impulso CLK8 hace que el contador inicie un nuevo ciclo. Para conseguir este modo de operación, se conecta Q_0 a las entradas J_1 y K_1 de FF1. Cuando Q_0 está a 1 y se produce un impulso de reloj, FF1 se encuentra en modo de basculación y, por tanto, cambia de estado. El resto de las veces, cuando Q_0 es 0, FF1 está en modo no cambio, quedando en su estado actual.

A continuación, vamos a ver cómo se consigue que FF2 cambie de estado en los instantes adecuados de acuerdo a la secuencia binaria. Obsérvese que las dos veces que Q_2 cambia de estado, debe cumplirse la única condición de que tanto Q_0 como Q_1 estén a nivel ALTO. Esta condición se detecta mediante la puerta AND, cuya salida se aplica a las entradas J_2 y K_2 de FF2. Siempre que Q_0 y Q_1 están a nivel ALTO, la salida de la puerta AND hace que las entradas J_2 y K_2 de FF2 se pongan a nivel ALTO, y FF2 bascula en el siguiente impulso de reloj. El resto de las veces, las entradas J_2 y K_2 de FF2 se mantienen a nivel BAJO, al igual que la salida de la puerta AND, y FF2 no cambia de estado.

NOTAS INFORMÁTICAS
El TSC (time stamp counter, contador de marca temporal) en el Pentium se utiliza para supervisar el funcionamiento, lo que permite determinar de forma exacta una serie de importantes parámetros dentro del funcionamiento global de un sistema Pentium. Leyendo el TSC antes y después de la ejecución de un procedimiento, se puede determinar el tiempo preciso requerido para el procedimiento, en función de la frecuencia de ciclo del procesador. De esta forma, el TSC es la base para determinar todas las temporizaciones relacionadas con la optimización del funcionamiento del sistema. Por ejemplo, se puede determinar de forma precisa cuál de dos o más secuencias de programación es la más eficiente. Ésta es una herramienta muy útil para los desarrolladores de compiladores y programadores de sistemas, a la hora de generar el código más eficiente para el Pentium.

Contadores síncronos binarios

La Figura 9.16(a) presenta un contador binario síncrono de 4 bits y la Figura 9.16(b) muestra su diagrama de tiempos. Este contador particular se implementa con flip-flops disparados por flanco negativo. El razonamiento para controlar las entradas J y K de los tres primeros flip-flops es el mismo que el del contador de 3 bits, previamente estudiado. La cuarta etapa, FF3, varía sólo dos veces en la secuencia. Obsérvese que estas dos transiciones ocurren justo cuando Q_0, Q_1 y Q_2 están a nivel ALTO. Esta condición se decodifica mediante la puerta AND G_2 de forma que, cuando se produce un impulso de reloj, FF3 cambia de estado. En los demás casos, las entradas J_3 y K_3 de FF3 están a nivel BAJO y se produce la condición de no cambio.

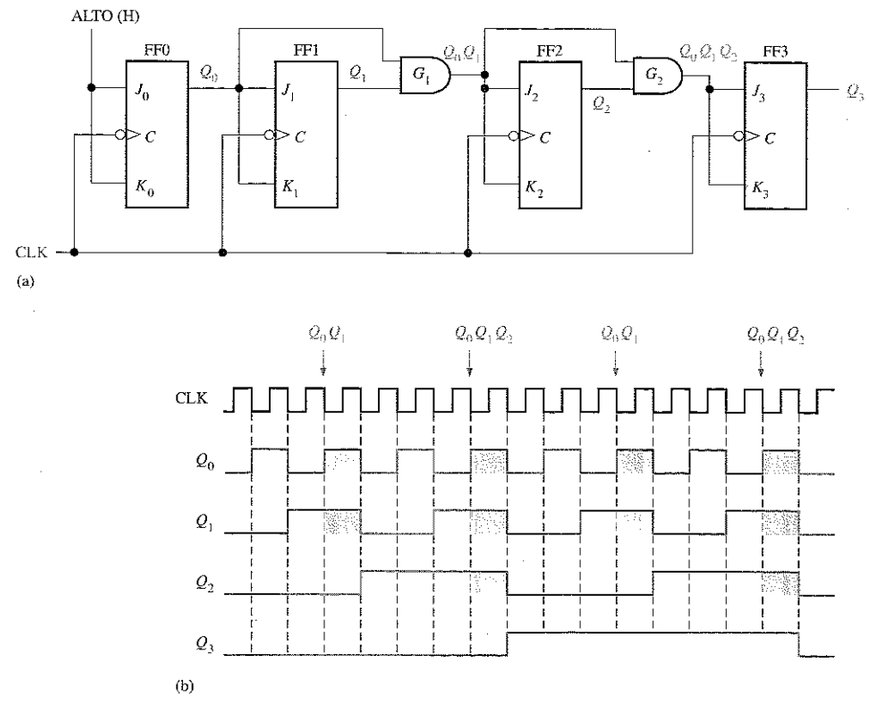


FIGURA 9.16
Contador binario síncrono de 4 bits y diagrama de tiempos. Los instantes en que las salidas de las puertas AND están a nivel ALTO se indican con áreas sombreadas.

Contador de décadas síncrono de 4 bits

Como ya sabemos, un contador de décadas BCD dispone de una secuencia binaria truncada que va desde 0000 hasta el estado 1001. En lugar de pasar al estado 1010, inicia un nuevo ciclo a partir del estado 0000. En la Figura 9.17 se presenta un contador de décadas BCD síncrono. En la Figura 9.18 se muestra el diagrama de tiempos para este contador de décadas.

El funcionamiento de este contador se puede entender examinando la secuencia de estados de la Tabla 9.4, y siguiendo la implementación de la Figura 9.17. En primer lugar, obsérvese que FF0 (Q_0) bascula en cada impulso de reloj, por lo que la ecuación lógica para sus entradas J_0 y K_0 es:

$$J_0 = K_0 = 1$$

Esta ecuación se implementa conectando J_0 y K_0 a un nivel ALTO constante.

A continuación, obsérvese en la Tabla 9.4 que FF1 (Q_1) cambia en el siguiente impulso de reloj cada vez que $Q_0 = 1$ y $Q_3 = 0$, por lo que la ecuación lógica para las entradas J_1 y K_1 es:

$$J_1 = K_1 = Q_0 \overline{Q}_3$$

Esta ecuación se implementa aplicando la operación AND a las salidas Q_0 y $\overline{Q}_3$, y conectando la salida de la puerta a las entradas J_1 y K_1 de FF1.

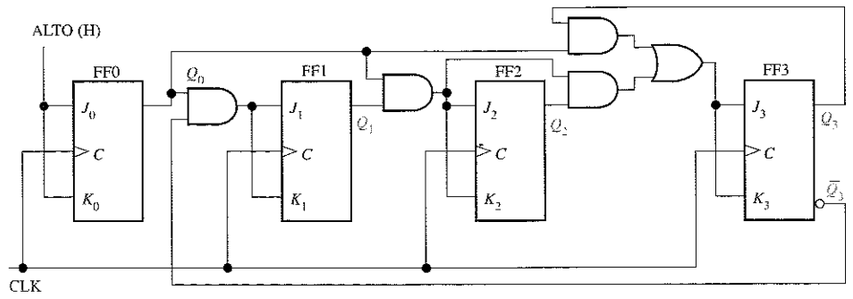


FIGURA 9.17
Contador de décadas BCD síncrono.

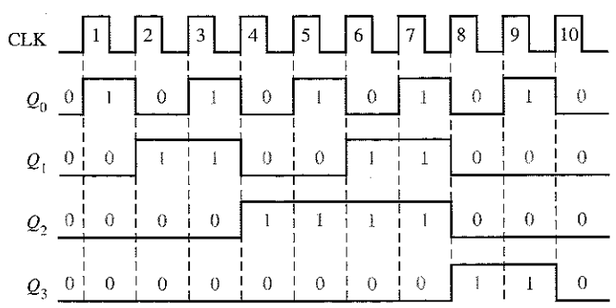


FIGURA 9.18
Cronograma del contador de décadas BCD (Q_0 es el LSB).

El flip-flop 2 (Q_2) cambia de estado en el siguiente impulso de reloj cada vez que $Q_0 = 1$ y $Q_1 = 1$. Luego la ecuación lógica de entrada es:

$$J_2 = K_2 = Q_0 Q_1$$

TABLA 9.4
Estados del contador de décadas BCD.

Pulso de reloj	Q_3	Q_2	Q_1	Q_0
Inicialización	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10 (nuevo ciclo)	0	0	0	0

Esta ecuación se implementa aplicando Q_0 y Q_1 a las entradas de una puerta AND, y conectando la salida de la puerta a las entradas J_2 y K_2 de FF2.

Finalmente, FF3 (Q_3) cambia de estado en el siguiente impulso de reloj cada vez que $Q_0 = 1$, $Q_1 = 1$ y $Q_2 = 1$ (estado 7), o cuando $Q_0 = 1$ y $Q_3 = 1$ (estado 9). La ecuación que rige esto es la siguiente:

$$J_3 = K_3 = Q_0 Q_1 Q_2 + Q_0 Q_3$$

Esta función se implementa mediante la lógica AND/OR conectada a las entradas J_3 y K_3 de FF3, como se muestra en el diagrama lógico de la Figura 9.17. Obsérvese que la única diferencia entre este contador de décadas y el contador binario de módulo 16 de la Figura 9.16 es la puerta AND con entradas $Q_0 Q_3$ y la puerta OR; esta disposición detecta la ocurrencia del estado 1001 y hace que el contador inicie un nuevo ciclo correctamente en el siguiente impulso de reloj.

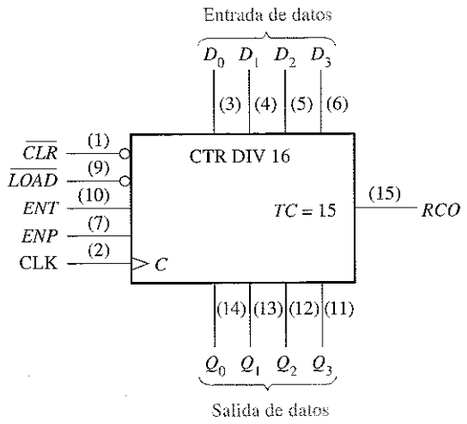
Circuito integrado contador binario síncrono de 4 bits

El 74HC163 es un ejemplo de un circuito integrado contador binario síncrono de 4 bits. El símbolo lógico se muestra en la Figura 9.19, con la numeración de pines entre paréntesis. Este contador tiene varias características adicionales con respecto a las características básicas del contador binario síncrono general previamente tratado.

En primer lugar, el contador puede reinicializarse de forma síncrona en cualquier número binario de 4 bits, aplicando los niveles adecuados en las entradas de datos paralelo. Cuando se aplica un nivel BAJO a la entrada $\overline{LOAD}$, el contador asumirá el estado de las entradas de datos en el siguiente impulso de reloj. Por tanto, la secuencia del contador se puede iniciar con cualquier número binario de 4 bits.

Además, hay una entrada de borrado activa a nivel BAJO ($\overline{CLR}$) que pone a cero de forma síncrona los cuatro flip-flops del contador. Hay dos entradas de habilitación, *ENP* y *ENT*. Estas entradas deben estar a nivel ALTO para que el contador pueda avanzar a través de su secuencia de estados binarios. Cuando al menos una de las entradas está a nivel BAJO, el contador se desactiva. La salida de propagación de reloj (ripple clock output, *RCO*) se pone a nivel ALTO cuando el contador alcanza el **valor de fin de cuenta** (terminal count, TC) de quince ($TC = 15$). Esta salida, junto con las entradas de habilitación permiten que estos contadores se puedan disponer en cascada para conseguir secuencias de cuenta mayores, como se verá más adelante.

FIGURA 9.19
El contador binario síncrono de 4 bits 74HC163. La etiqueta CTR DIV 16 indica un contador con dieciséis estados.



La Figura 9.20 muestra un cronograma de este contador, que se inicializa en el estado 12 (1100) y luego avanza hasta su valor de fin de cuenta 15 (1111). La entrada D_0 corresponde al bit de entrada menos significativo y Q_0 es el bit de salida menos significativo.

Vamos a examinar este cronograma en detalle. Esto nos ayudará a interpretar los diagramas de tiempos que encontraremos más adelante en este mismo capítulo o en las hojas de características de los fabricantes. Para comenzar, el impulso a nivel BAJO en la entrada $\overline{CLR}$ hace que todas las salidas (Q_0, Q_1, Q_2 y Q_3) se pongan a nivel BAJO.

A continuación, el impulso a nivel BAJO en la entrada $\overline{LOAD}$ introduce de forma síncrona los datos en las entradas (D_0, D_1, D_2 y D_3) del contador. Éstos aparecen en las salidas Q cuando se produce el primer flanco positivo de reloj después de que $\overline{LOAD}$ pasa a nivel BAJO. Esta es la operación de inicialización (PRESET). En este ejemplo particular, Q_0 está a nivel BAJO, Q_1 está a nivel BAJO, Q_2 está a nivel ALTO y Q_3 está a nivel ALTO. Por supuesto, esto corresponde al número decimal 12 (Q_0 es el bit menos significativo).

Ahora, el contador avanza por los estados 13, 14 y 15 en los tres siguientes flancos positivos de reloj, y luego comienza un nuevo ciclo en 0, 1, 2 con los siguientes impulsos de reloj. Obsérvese que las dos entradas *ENP* y *ENT* están a nivel ALTO durante la secuencia de estados. Cuando *ENP* o *ENT* pasan a nivel BAJO, el contador se inhibe y permanece en el estado binario 2.

Un contador de décadas BCD síncrono

El 74HC160 tiene las mismas entradas y salidas que el contador binario 74HC163 estudiado previamente. Se puede inicializar con cualquier número BCD utilizando las entradas de datos con la entrada $\overline{LOAD}$ a nivel BAJO. Un nivel BAJO en $\overline{CLR}$ pone en estado RESET al contador. Las entradas de habilitación *ENP* y *ENT* tienen que estar ambas a nivel ALTO para que el contador avance a través de la secuencia de estados, en respuesta a una transición positiva en la entrada de reloj CLK. Al igual que en el 74HC163, las entradas de habilitación junto con la salida de propagación de reloj

RCO (valor de fin de cuenta de 1001), permiten conectar varios contadores de décadas en cascada. La Figura 9.21 nos muestra el símbolo lógico del contador 74HC160, y la Figura 9.22 presenta un diagrama de tiempos del contador inicializado en el estado 7 (0111). Los contadores en cascada se tratarán en la Sección 9.5.

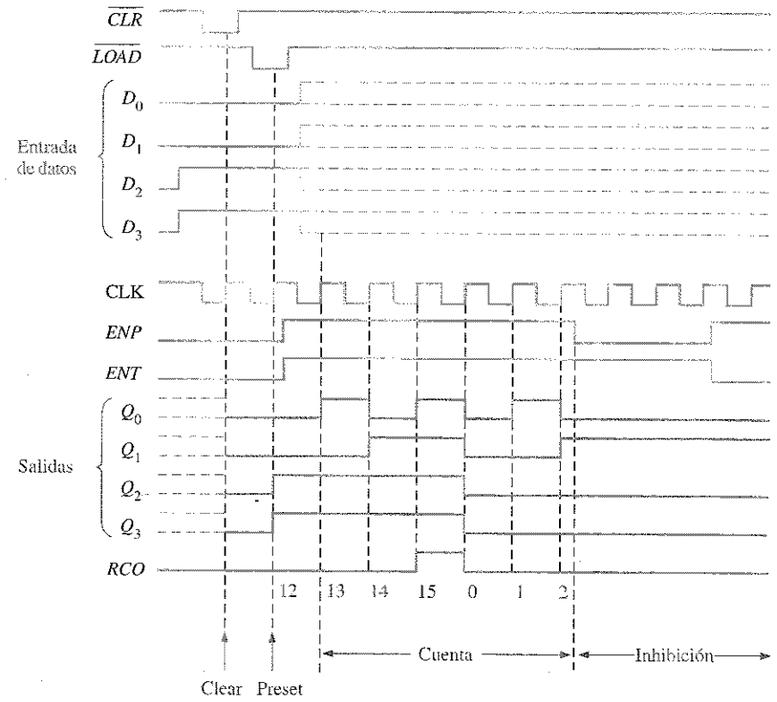


FIGURA 9.20
Ejemplo de un diagrama de tiempos para un 74HC163.

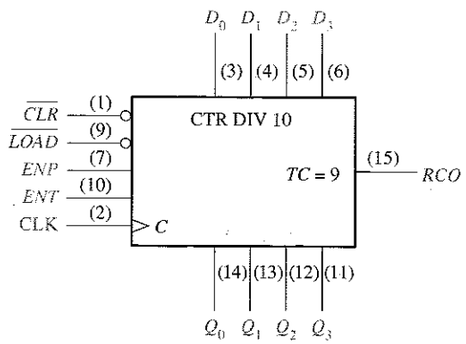


FIGURA 9.21
El contador de décadas BCD síncrono 74HC160. La etiqueta CTR DIV 10 indica un contador con diez estados.

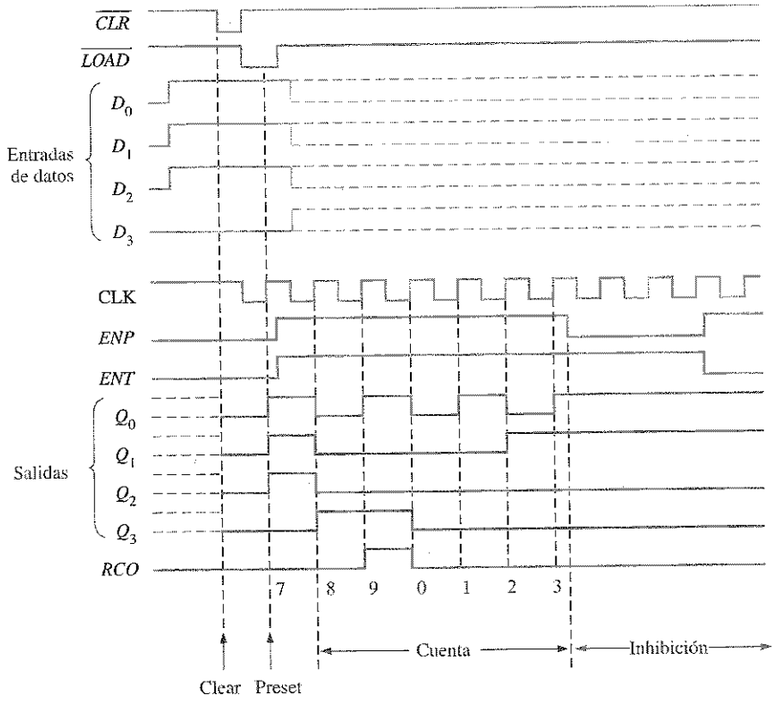


FIGURA 9.22
Ejemplo de diagrama de tiempos de un 74HC160.

REVISIÓN DE LA SECCIÓN 9.2

1. ¿En qué se diferencia un contador síncrono de uno asíncrono?
2. Explicar el funcionamiento de la función preset de contadores tales como el 74HC160 y el 74HC163.
3. Describir el propósito de las entradas *ENP* y *ENT*, y de la salida *RCO* para los dos contadores específicos presentados en esta sección.

9.3 ■ CONTADOR SÍNCRONO ASCENDENTE/DESCENDENTE

Un contador ascendente/descendente (up/down) es aquel capaz de progresar en cualquier dirección a lo largo de una cierta secuencia. Un contador ascendente/descendente, algunas veces también denominado contador bidireccional, puede tener cualquier secuencia de estados especificada. Un contador binario de 3 bits que avanza en modo ascendente a través de la secuencia (0, 1, 2, 3, 4, 5, 6, 7) y que luego puede invertirse para recorrer la secuencia en sentido contrario (7, 6, 5, 4, 3, 2, 1, 0) es un ejemplo de un modo de operación secuencial ascendente/descendente.

Al finalizar esta sección, el lector deberá ser capaz de:

- Explicar el funcionamiento básico de un contador ascendente/descendente.
- Utilizar el contador de décadas ascendente/descendente 74LS190.

En general, la mayoría de los **contadores ascendentes/descendentes** pueden invertirse en cualquier punto de su secuencia. Por ejemplo, el contador binario de 3 bits se puede configurar para que realice la siguiente secuencia:

$$\overbrace{0, 1, 2, 3, 4, 5}^{\text{ASCENDENTE}}, \underbrace{4, 3, 2}_{\text{DESCENDENTE}}, \overbrace{3, 4, 5, 6, 7}^{\text{ASCENDENTE}}, \underbrace{6, 5}_{\text{DESCENDENTE}}, \text{etc.}$$

La Tabla 9.5 muestra la secuencia ascendente/descendente (up/down) completa de un contador binario de 3 bits. Las flechas indican los movimientos entre los estados del contador, tanto para el modo ascendente como para el modo descendente. Un examen de Q_0 para ambas secuencias, ascendente y descendente, muestra que FF0 bascula con cada impulso de reloj. Luego las entradas J_0 y K_0 de FF0 son:

$$J_0 = K_0 = 1$$

Para la secuencia ascendente, Q_1 cambia de estado en el siguiente impulso de reloj cuando $Q_0 = 1$. Para la secuencia descendente, Q_1 cambia en el siguiente impulso de reloj cuando $Q_0 = 0$. Por tanto, las entradas J_1 y K_1 de FF1 tienen que ser igual a 1, para las condiciones expresadas en la siguiente ecuación:

$$J_1 = K_1 = (Q_0 \cdot \text{UP}) + (\overline{Q}_0 \cdot \text{DOWN})$$

Para la secuencia ascendente, Q_2 cambia de estado en el siguiente impulso de reloj cuando $Q_0 = Q_1 = 1$. Para la secuencia descendente, Q_2 cambia en el siguiente impulso de reloj cuando $Q_0 = Q_1 = 0$. Por tanto, las entradas J_2 y K_2 de FF2 tienen que ser igual a 1, para las condiciones expresadas en la siguiente ecuación:

$$J_2 = K_2 = (Q_0 \cdot Q_1 \cdot \text{UP}) + (\overline{Q}_0 \cdot \overline{Q}_1 \cdot \text{DOWN})$$

Cada una de las condiciones para las entradas J y K de cada flip-flop produce una basculación en el punto apropiado de la secuencia del contador.

TABLA 9.5
Secuencia ascendente/descendente de un contador binario de 3 bits.

Pulso de reloj	Ascendente (UP)	Q_2	Q_1	Q_0	Descendente (DOWN)
0	→	0	0	0	
1		0	0	1	
2		0	1	0	
3		0	1	1	
4		1	0	0	
5		1	0	1	
6		1	1	0	
7		1	1	1	←

La Figura 9.23 muestra una implementación básica de un contador binario de 3 bits ascendente/ descendente, utilizando las ecuaciones lógicas que acabamos de desarrollar para las entradas J y K de cada flip-flop. Obsérvese que la entrada de control $\overline{UP/DOWN}$ *(ascendente/descendente)* está a nivel ALTO cuando trabaja en modo ascendente y a nivel BAJO cuando trabaja en modo descendente.

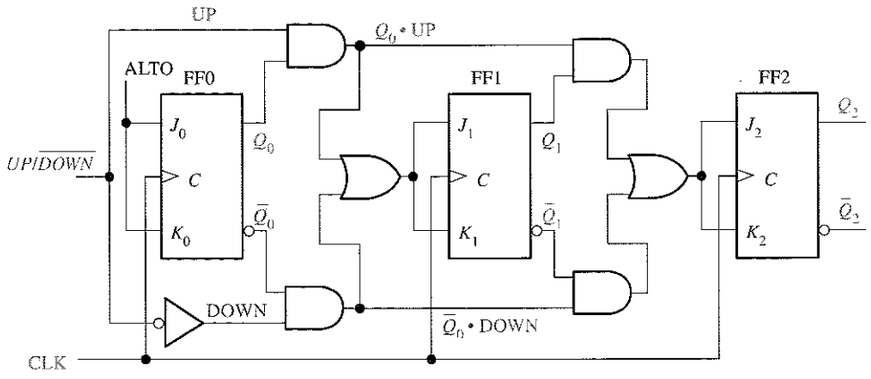


FIGURA 9.23
Contador síncrono ascendente/descendente básico de 3 bits.

EJEMPLO 9.4 Dibujar el diagrama de tiempos y determinar la secuencia de un contador síncrono binario de 4 bits ascendente/descendente, si el reloj y las entradas de control *UP/*$\overline{DOWN}$ son las señales que se muestran en la Figura 9.24(a). El contador se inicializa en el estado cero y es de tipo disparado por flanco positivo.

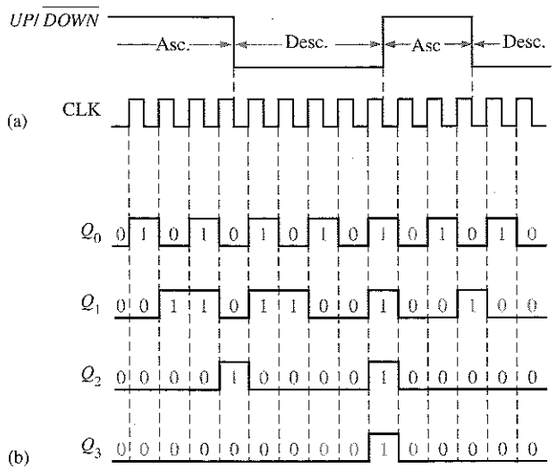


FIGURA 9.24

Solución. En la Figura 9.24(b) se presenta el diagrama de tiempos, mostrando las salidas Q. A partir de estas formas de onda, la secuencia del contador es la que se indica en la Tabla 9.6.

TABLA 9.6

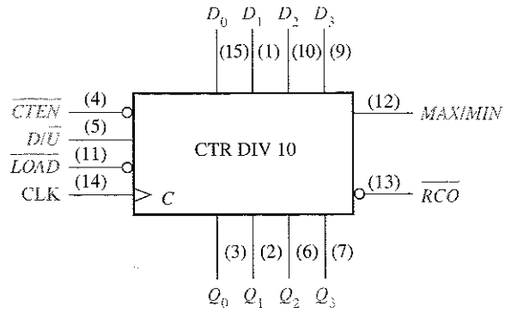
Q_3	Q_2	Q_1	Q_0	
0	0	0	0	ASCENDENTE
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	0	1	1	DESCENDENTE
0	0	1	0	
0	0	0	1	
0	0	0	0	
1	1	1	1	
0	0	0	0	ASCENDENTE
0	0	0	1	
0	0	1	0	
0	0	0	1	DESCENDENTE
0	0	0	0	

Problema relacionado. Dibujar el diagrama de tiempos si se invierte la señal de control *UP/*$\overline{DOWN}$ de la Figura 9.24(a).

Un contador de décadas ascendente/descendente

La Figura 9.25 muestra el diagrama lógico del 74LS190, buen ejemplo de un circuito integrado contador ascendente/descendente. La dirección de la cuenta se determina por el nivel de la entrada up/down ($D/\overline{U}$). Cuando esta entrada está a nivel ALTO, el contador se decrementa (desciende); cuando está a nivel BAJO, el contador se incrementa (asciende). Además, este dispositivo se puede inicializar en cualquier dígito BCD que se desee, el cual se carga a través de las entradas de datos cuando la entrada $\overline{LOAD}$ está a nivel BAJO.

FIGURA 9.25
El contador de décadas ascendente/descendente síncrono 74LS190



La salida *MAX/MIN* produce un impulso a nivel ALTO cuando se alcanza el valor de fin de cuenta de nueve (1001) en el modo ascendente, o cuando se alcanza el

valor de fin de cuenta de cero (0000) en el modo descendente. La salida *MAX/MIN*, junto con la salida de propagación de reloj ($\overline{RCO}$) y la entrada de habilitación de cuenta (*CTEN*), se usa para conectar contadores en cascada (los contadores en cascada se verán en la Sección 9.5).

La Figura 9.26 es un diagrama de tiempos ejemplo, que muestra un contador 74LS190 inicializado en siete (0111); el contador luego describe una secuencia ascendente, seguida de una secuencia descendente. La salida *MAX/MIN* está a nivel ALTO cuando el contador está en el estado 0 (*MIN*) o en el estado 1001 (*MAX*).

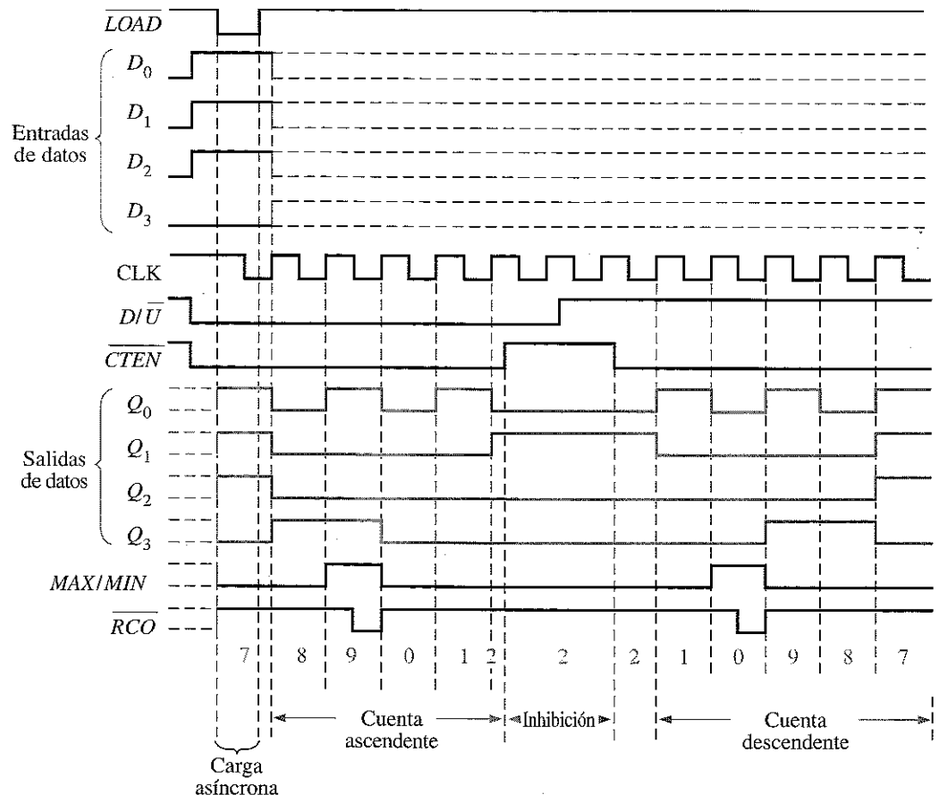


FIGURA 9.26
Ejemplo de un diagrama de tiempos para un 74LS190.

REVISIÓN DE LA SECCIÓN 9.3

1. Un contador binario de 4 bits ascendente/descendente se encuentra en modo descendente y en el estado 1010. En el siguiente impulso de reloj, ¿a qué estado pasa?
2. ¿Cuál es el valor de fin de cuenta de un contador binario de 4 bits en modo ascendente? ¿Y en modo descendente? ¿Cuál es el siguiente estado después del valor final de cuenta en modo descendente?

9.4 ■ DISEÑO DE CONTADORES SÍNCRONOS*

Esta sección es opcional y puede ser omitida sin que afecte al material expuesto en lo que resta de libro. Sin embargo, se recomienda su estudio a aquellos que deseen una introducción al diseño de contadores o de máquinas de estados en general.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir un circuito secuencial general en función de sus partes básicas, y de sus entradas y salidas. ■ Desarrollar un diagrama de estados para una determinada secuencia. ■ Desarrollar una tabla del estado siguiente para una secuencia de contador específica. ■ Crear una tabla de transiciones de flip-flops. ■ Utilizar el método del mapa de Karnaugh para obtener los requisitos lógicos de un contador asíncrono. ■ Implementar un contador para producir una secuencia de estados específica.

Modelo general de un circuito secuencial

Antes de exponer una técnica de diseño de contadores específica, vamos a comenzar con una definición general de **circuito secuencial (máquina de estados)**: un circuito secuencial está formado por una etapa de lógica combinacional y una etapa de memoria (flip-flops), como se muestra en la Figura 9.27. En un circuito secuencial sincronizado, hay una entrada de reloj en la etapa de memoria, tal como se indica.

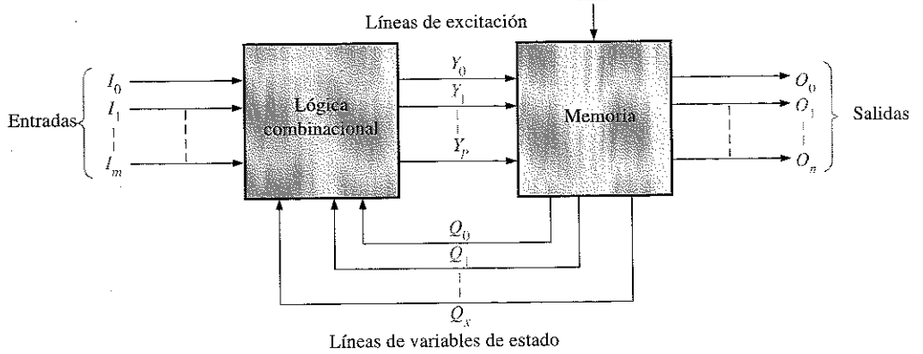


FIGURA 9.27
Circuito secuencial general sincronizado.

Para el correcto funcionamiento del circuito se requiere la información almacenada en la etapa de memoria, así como las entradas de la lógica combinacional (I_0, I_1, ..., I_m). En cualquier instante de tiempo, la memoria se encuentra en un estado denominado *estado actual* y avanza al *estado siguiente* con un impulso de reloj determinado por las condiciones de las líneas de excitación (Y_0, Y_1, ..., Y_p). El estado actual de la memoria se representa por las variables de estado (Q_0, Q_1, ..., Q_x). Estas variables de estado, junto con las entradas (I_0, I_1, ..., I_m), determinan las salidas del sistema (O_0, O_1, ..., O_n).

* Esta sección es opcional y puede omitirse sin que ello afecte a los temas tratados en el resto del libro.

No todos los circuitos secuenciales tienen variables de entrada y salida como en el modelo general que se acaba de presentar. Sin embargo, todos tienen variables de excitación y variables de estado. Los contadores son un caso particular de los circuitos secuenciales sincronizados. En esta sección, se aplica un procedimiento de diseño general de los circuitos secuenciales a los contadores síncronos a través de una serie de pasos.

Paso 1: diagrama de estados

En primer lugar, se describe el contador mediante un **diagrama de estados**, que muestra la progresión de estados por los que el contador avanza cuando se aplica una señal de reloj. Como ejemplo, en la Figura 9.28, se muestra un diagrama de estados de un contador básico en código Gray de 3 bits. Este circuito particular no tiene ninguna entrada aparte de la de reloj, y ninguna otra salida más que las que se toman en cada flip-flop del contador. Si lo desea, puede repasar el código Gray, descrito en el Capítulo 2.

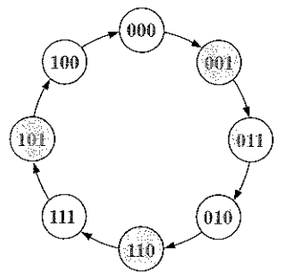


FIGURA 9.28
Diagrama de estados para un contador en código Gray de 3 bits.

Paso 2: tabla del estado siguiente

Una vez que se define el circuito secuencial mediante un diagrama de estados, el segundo paso consiste en obtener una tabla del estado siguiente, que enumera cada estado del contador (estado actual) junto con el correspondiente estado siguiente. *El estado siguiente es el estado al que el contador pasa desde su estado actual, al aplicar un impulso de reloj*. La tabla del estado siguiente se obtiene a partir del diagrama de estados, y se muestra en la Tabla 9.7 para el contador en código Gray de 3 bits. Q_0 es el bit menos significativo.

Paso 3: tabla de transiciones de los flip-flops

La Tabla 9.8 es una tabla de transiciones del flip-flop J-K. Se enumeran todas las posibles transiciones de salida, mostrando cómo evoluciona la salida Q del flip-flop al pasar de los estados actuales a los estados siguientes. Q_N es el estado presente en el flip-flop (antes de un impulso de reloj) y Q_{N+1} es el estado siguiente (después de un

impulso de reloj). Para cada transición de salida, se indican las entradas J y K que dan lugar a la transición. Las "X" indican condiciones indiferentes (la entrada puede ser un 1 o un 0).

TABLA 9.7
Tabla del estado siguiente para el contador en código Gray de 3 bits.

Estado actual			Estado siguiente		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	0	0	1
0	0	1	0	1	1
0	1	1	0	1	0
0	1	0	1	1	0
1	1	0	1	1	1
1	1	1	1	0	1
1	0	1	1	0	0
1	0	0	0	0	0

Al diseñar el contador, se aplica la tabla de transiciones a cada flip-flop del contador, la cual está basada en la tabla del estado siguiente (Tabla 9.7). Por ejemplo, para el estado actual 000, Q_0 pasa del estado actual 0 al estado siguiente 1. Para que esto ocurra, J_0 tiene que ser 1 y es indiferente el valor que tome K_0 ($J_0 = 1$, $K_0 = X$), como se indica en la tabla de transiciones (Tabla 9.8). A continuación, el estado actual de Q_1 es 0 y permanece en 0 en el estado siguiente. Para esta transición, $J_1 = 0$ y $K_1 = X$. Por último, el estado actual de Q_2 es 0 y permanece en 0 en el estado siguiente. Por tanto, $J_2 = 0$ y $K_2 = X$. Este análisis se repite para cada estado actual definido en la Tabla 9.7.

TABLA 9.8
Tabla de transiciones para un flip-flop J-K.

Transiciones de salida			Entradas del flip-flop	
Q_N		Q_{N+1}	J	K
0	⟶	0	0	X
0	⟶	1	1	X
1	⟶	0	X	1
1	⟶	1	X	0

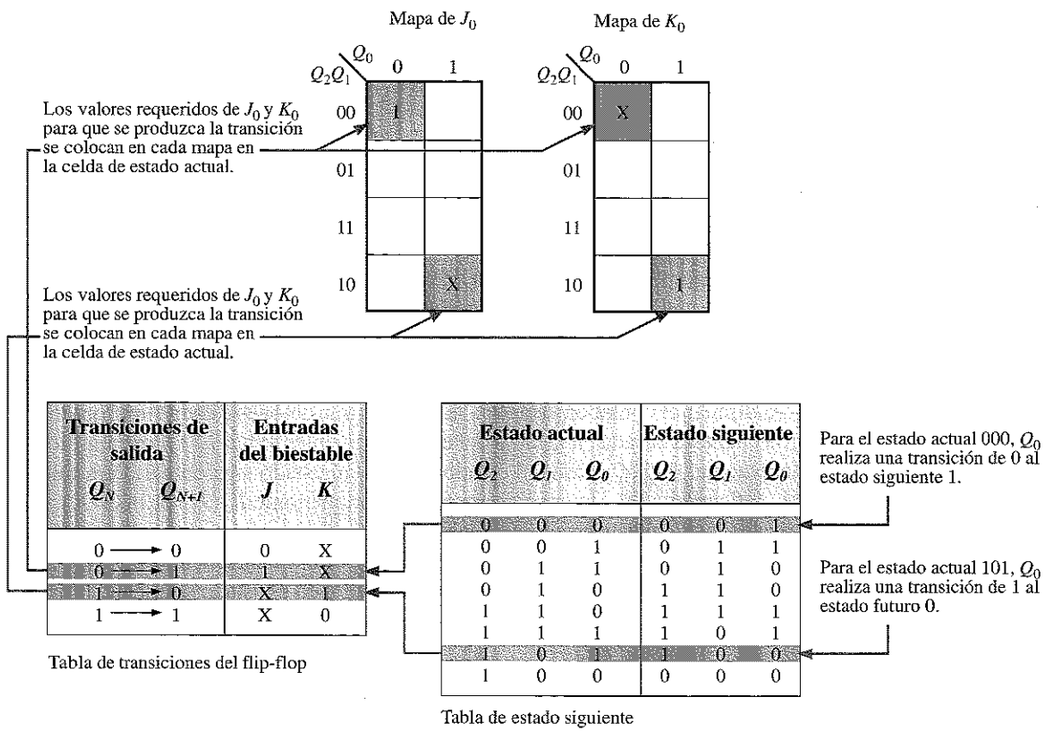
Q_N: estado actual
Q_{N+1}: estado siguiente
X: indiferente

Paso 4: mapas de Karnaugh

Los mapas de Karnaugh se utilizan para determinar la lógica requerida para las entradas J y K de cada flip-flop del contador. Se debe utilizar un mapa de Karnaugh para la entrada J y otro para la entrada K de cada flip-flop. En este procedimiento de diseño, cada celda del mapa de Karnaugh representa uno de los estados actuales de la secuencia del contador enumerados en la Tabla 9.7.

A partir de los estados J y K de la tabla de transiciones (Tabla 9.8) se introduce un 1, un 0 o una X en cada celda de la tabla correspondiente al estado actual, dependiendo de la transición de la salida Q de cada flip-flop en particular. Para ilustrar este

procedimiento, se muestran en la Figura 9.29 dos valores de entrada de ejemplo para las entradas J_0 y K_0 del flip-flop menos significativo (Q_0).



Transiciones de salida Q_N	Q_{N+1}	Entradas del biestable J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Estado actual Q_2	Q_1	Q_0	Estado siguiente Q_2	Q_1	Q_0
0	0	0	0	0	1
0	0	1	0	1	1
0	1	1	0	1	0
0	1	0	1	1	0
1	1	0	1	1	1
1	1	1	1	0	1
1	0	1	1	0	0
1	0	0	0	0	0

FIGURA 9.29
Ejemplos del procedimiento de utilización de mapas de Karnaugh para la secuencia de contador representada en las Tablas 9.7 y 9.8.

El mapa de Karnaugh completo de los tres flip-flops del contador se muestra en la Figura 9.30. Las celdas se agrupan tal como se indica, obteniéndose las expresiones booleanas correspondientes para cada grupo.

Paso 5: expresiones lógicas para las entradas de los flip-flops

A partir de los mapas de Karnaugh de la Figura 9.30 se obtienen las siguientes expresiones para las entradas J y K de cada flip-flop:

$$J_0 = Q_2Q_1 + \overline{Q}_2\overline{Q}_1 = \overline{Q_2 \oplus Q_1}$$
$$K_0 = Q_2\overline{Q}_1 + \overline{Q}_2Q_1 = Q_2 \oplus Q_1$$
$$J_1 = \overline{Q}_2Q_0$$
$$K_1 = Q_2Q_0$$
$$J_2 = Q_1\overline{Q}_0$$
$$K_2 = \overline{Q}_1\overline{Q}_0$$

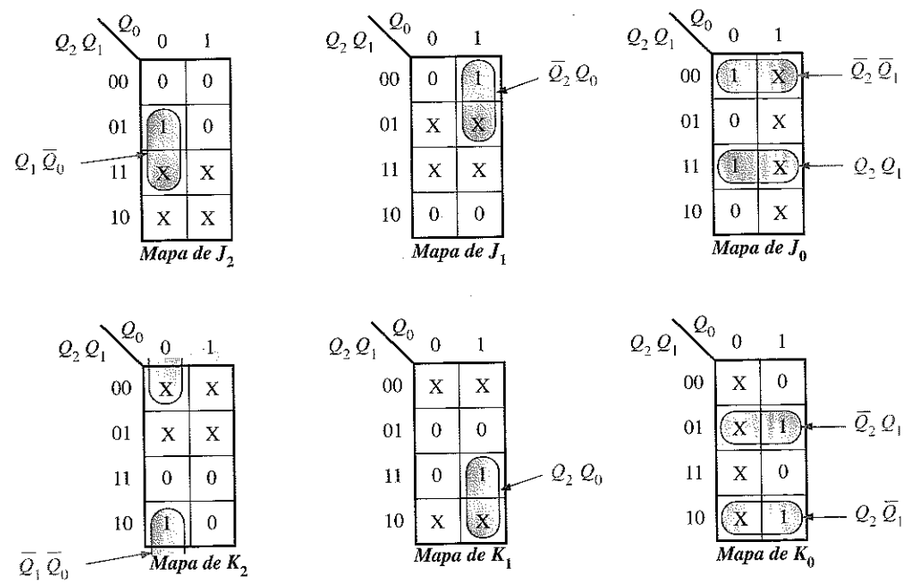


FIGURA 9.30
Mapas de Karnaugh para las entradas J y K del estado actual.

Paso 6: implementación del contador

El paso final consiste en implementar la lógica combinacional a partir de las expresiones de las entradas J y K, y conectar los flip-flops para conseguir un contador en código Gray de 3 bits, como se muestra en la Figura 9.31.

A continuación, se expone un resumen de los pasos dados en el diseño de este contador. En general, estos pasos se pueden aplicar a cualquier circuito secuencial.

1. Especificar la secuencia del contador y dibujar un diagrama de estados.
2. Obtener la tabla del estado siguiente a partir del diagrama de estados.
3. Desarrollar una tabla de transiciones que muestre las entradas del flip-flop requeridas para cada transición. La tabla de transiciones es siempre la misma para cada tipo de flip-flop.
4. Transferir los estados J y K de la tabla de transiciones al mapa de Karnaugh. Utilizar un mapa de Karnaugh para cada entrada de cada flip-flop.
5. Formar los términos productos a partir de los mapas para generar una expresión lógica, para cada entrada de los flip-flops.
6. Implementar la expresión con lógica combinacional y conectarla a los flip-flops para crear el contador.

A continuación, en los ejemplos 9.5 y 9.6, se va a aplicar este procedimiento al diseño de otros contadores síncronos.

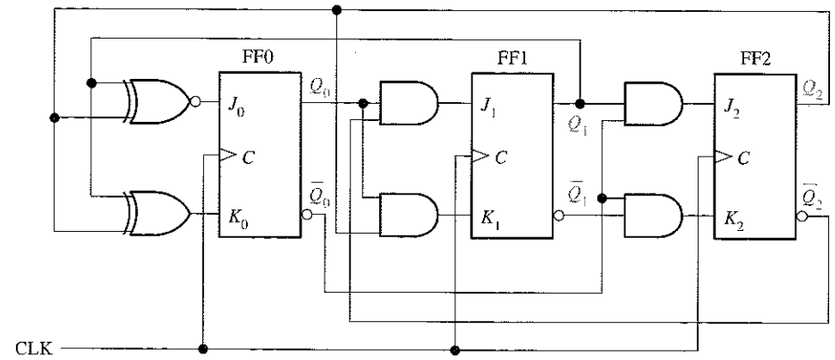


FIGURA 9.31
Contador en código de Gray de 3 bits.

EJEMPLO 9.5 Diseñar un contador que realice la secuencia de cuenta binaria irregular que se muestra en el diagrama de estados de la Figura 9.32. Utilizar flip-flops J-K.

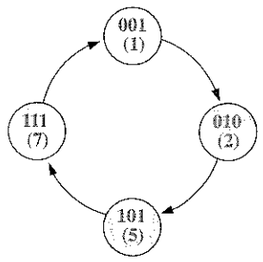


FIGURA 9.32

Solución

Paso 1. El diagrama de estados es el que se muestra en la figura. Aunque hay sólo cuatro estados, necesitamos un contador de 3 bits para implementar esta secuencia, dado que el número binario máximo es siete. Ya que la secuencia requerida no incluye todos los posibles estados binarios, los estados no válidos (0, 3, 4 y 6) pueden ser considerados como indiferentes en el diseño. Sin embargo, si el contador pasara por error por un estado no válido, debe asegurarse que luego volverá a un estado válido.

Paso 2. La tabla del estado siguiente se desarrolla a partir del diagrama de estados y se muestra en la Tabla 9.9.

Paso 3. En la Tabla 9.10 se repite la tabla de transiciones del flip-flop J-K.

Paso 4. En la Figura 9.33 se muestran las entradas J y K en los mapas de Karnaugh del estado actual. También se pueden incluir condiciones indiferentes (X) en las celdas correspondientes a los estados no válidos 000, 011, 100 y 110.

TABLA 9.9
Tabla del estado siguiente.

Estado actual			Estado siguiente		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	1	0	1	0
0	1	0	1	0	1
1	0	1	1	1	1
1	1	1	0	0	1

TABLA 9.10
Tabla de transiciones del flip-flop J-K.

Transiciones de salida		Entradas del flip-flop	
Q_N	Q_{N+1}	J	K
0 ⟶	0	0	X
0 ⟶	1	1	X
1 ⟶	0	X	1
1 ⟶	1	X	0

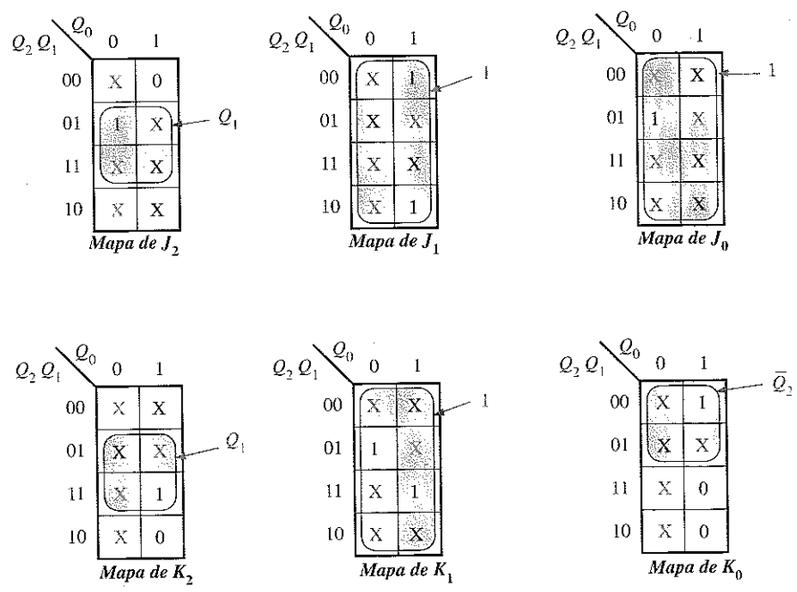


FIGURA 9.33

Paso 5. Agrupar los 1s, utilizando tantos estados indiferentes como sea posible para obtener la máxima simplificación, como se muestra en la Figura 9.33. Obsérvese que cuando se agrupan *todas* las celdas de una tabla, la expresión es simplemente 1. La expresión para cada entrada J y K a partir de los mapas es la que sigue:

$$J_0 = 1, K_0 = \overline{Q}_2$$
$$J_1 = K_1 = 1$$
$$J_2 = K_2 = Q_1$$

Paso 6. La implementación del contador se puede ver en la Figura 9.34.

Un análisis nos muestra que si, por accidente, el contador pasa a un estado no válido (0, 3, 4, 6) va a retornar siempre a un estado válido de acuerdo con las siguientes secuencias : $0 \rightarrow 3 \rightarrow 4 \rightarrow 7$ y $6 \rightarrow 1$.

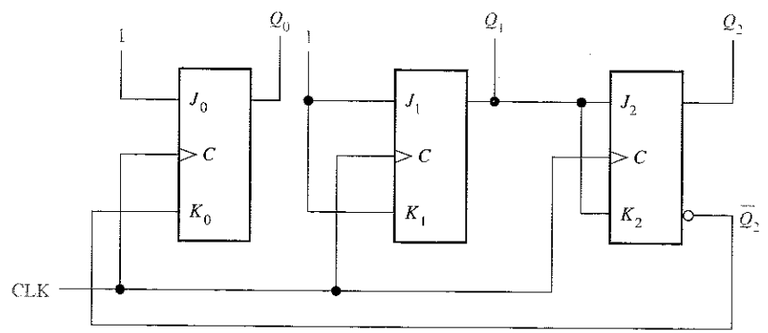


FIGURA 9.34

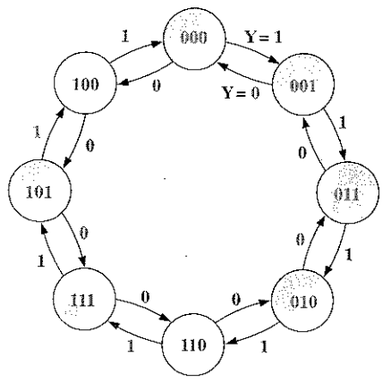
Problema relacionado. Verificar el análisis que demuestra que el contador (al final) siempre va a pasar a un estado válido desde un estado no válido.

EJEMPLO 9.6 Desarrollar un contador síncrono ascendente/descendente de 3 bits con una secuencia en código Gray. El contador trabajará en modo ascendente cuando la entrada de control UP/$\overline{\text{DOWN}}$ sea 1, y trabajará en modo descendente cuando la entrada de control sea 0.

Solución

Paso 1. El diagrama de estados se muestra en la Figura 9.35. El 1 o 0 al lado de cada flecha indica el estado de la entrada de control UP/$\overline{\text{DOWN}}$, Y.

FIGURA 9.35
Diagrama de estados de un contador en código Gray ascendente/ descendente de 3bits.



Paso 2. La tabla del estado siguiente se obtiene a partir del diagrama de estados y se presenta en la Tabla 9.11. Obsérvese que para cada estado actual hay dos posibles estados siguientes, dependiendo de la variable UP/$\overline{\text{DOWN}}$ de control, Y.

Paso 3. La tabla de transiciones para el flip-flop J-K se repite en la Tabla 9.12.

Paso 4. Los mapas de Karnaugh para las entradas J y K de los flip-flops se presentan en la Figura 9.36. La entrada de control UP/$\overline{\text{DOWN}}$, Y, se considera una de las variables de estado junto con Q_0, Q_1 y Q_2. Utilizando la tabla del estado siguiente, la información de la columna "Entradas del flip-flop" de la Tabla 9.12 se transfiere a las tablas indicadas para cada estado actual del contador.

TABLA 9.11
Tabla del estado siguiente del contador en código Gray ascendente/ descendente de 3 bits.

Estado actual			Estado siguiente $Y = 0$ (DOWN)			Estado siguiente $Y = 1$ (UP)		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1	1
0	1	1	0	0	1	0	1	0
0	1	0	0	1	1	1	1	0
1	1	0	0	1	0	1	1	1
1	1	1	1	1	0	1	0	1
1	0	1	1	1	1	1	0	0
1	0	0	1	0	1	0	0	0

Y = Entrada de control UP/$\overline{\text{DOWN}}$.

TABLA 9.12
Tabla de transiciones del flip-flop J-K.

Transiciones de salida		Entradas del flip-flop	
Q_N	Q_{N+1}	J	K
0 ⟶	0	0	X
0 ⟶	1	1	X
1 ⟶	0	X	1
1 ⟶	1	X	0

Paso 5. Los 1s se combinan formando grupos tan grandes como sea posible, utilizando las condiciones indiferentes (X) siempre que se pueda. Se forman los términos producto y se obtienen las expresiones siguientes para las entradas J y K:

$$J_0 = Q_2Q_1Y + Q_2\overline{Q}_1\overline{Y} + \overline{Q}_2Q_1\overline{Y} + \overline{Q}_2\overline{Q}_1Y \qquad K_0 = \overline{Q}_2\overline{Q}_1\overline{Y} + \overline{Q}_2Q_1Y + Q_2Q_1\overline{Y} + Q_2\overline{Q}_1Y$$

$$J_1 = \overline{Q}_2Q_0Y + Q_2Q_0\overline{Y} \qquad K_1 = \overline{Q}_2Q_0\overline{Y} + Q_2Q_0Y$$

$$J_2 = Q_1\overline{Q}_0Y + \overline{Q}_1\overline{Q}_0\overline{Y} \qquad K_2 = Q_1\overline{Q}_0\overline{Y} + \overline{Q}_1\overline{Q}_0Y$$

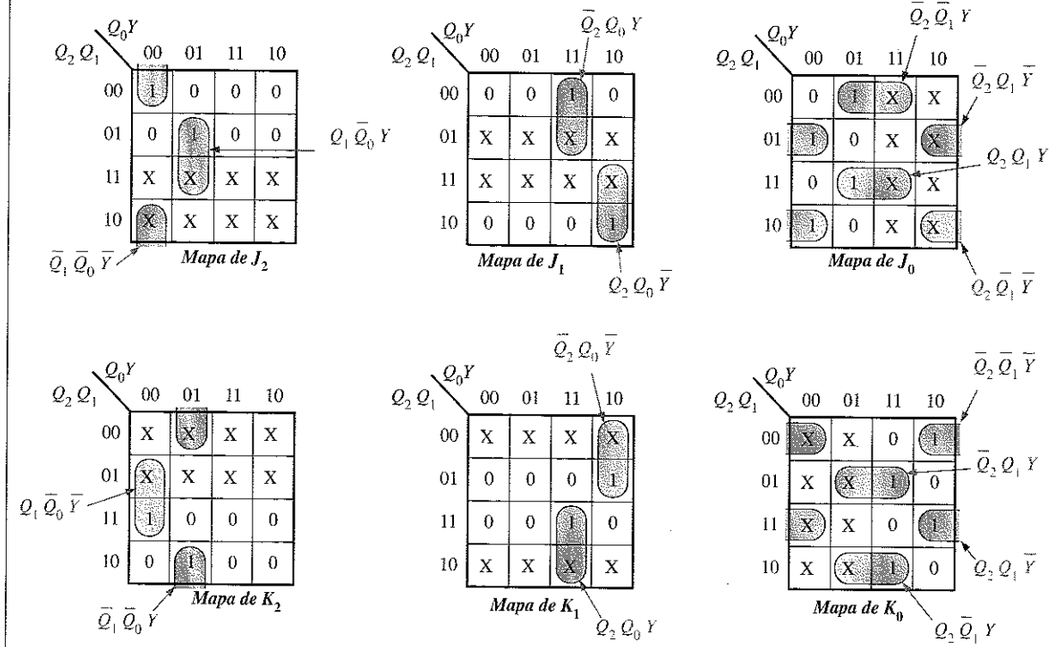


FIGURA 9.36
Mapas para J y K de acuerdo con la Tabla 9.11. La entrada de control UP/DOWN, Y, se trata como una cuarta variable.

Paso 6. Las ecuaciones de J y K se implementan con lógica combinacional, obteniendo el contador completo que se muestra en la Figura 9.37. Éste es un buen candidato para implementación en un PLD.

Problema relacionado. Verificar que la lógica de la Figura 9.37 concuerda con las expresiones del paso 5.

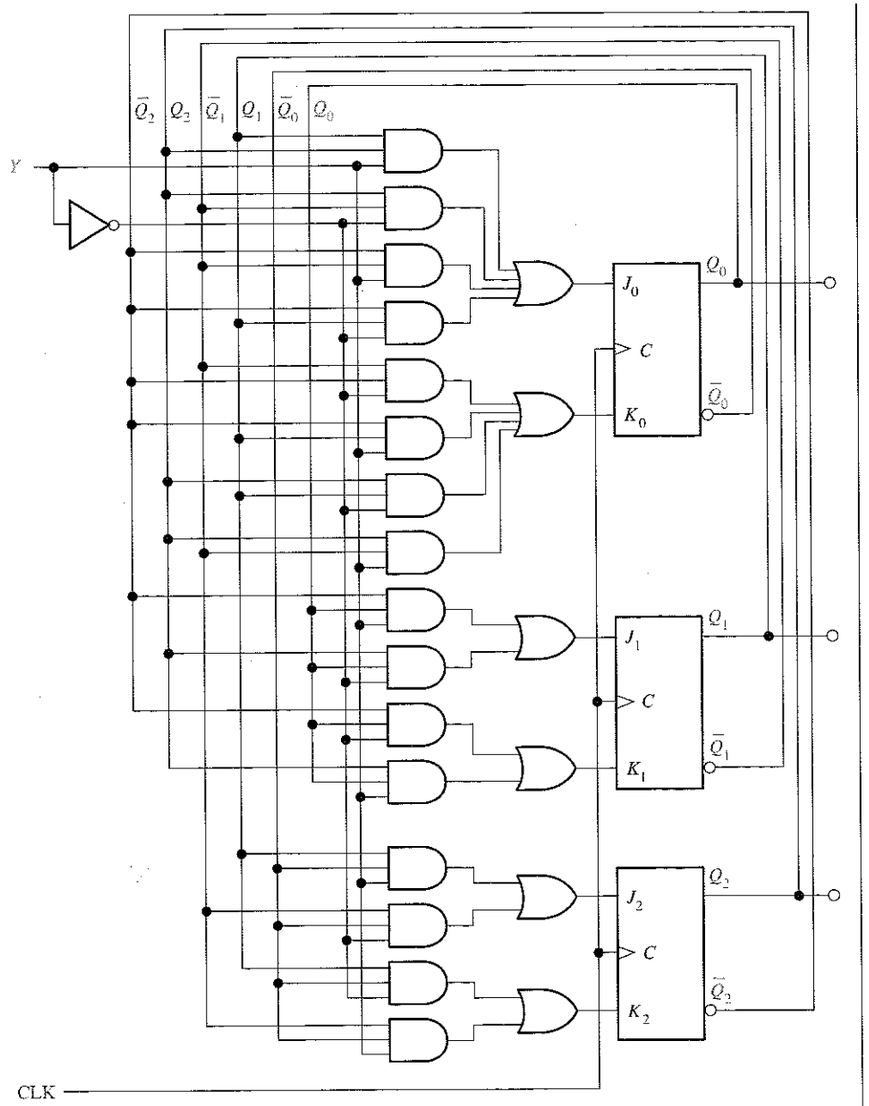


FIGURA 9.37
Contador ascendente/descendente en código Gray de tres bits.

En esta sección, se ha visto cómo las técnicas de diseño de circuitos secuenciales se pueden aplicar al diseño de contadores. Por lo general, los circuitos secuenciales se pueden clasificar en dos tipos: (1) aquellos en los que la salida o salidas dependen sólo del estado actual interno (*circuitos Moore*), y, (2) aquellos en los que la salida o salidas dependen tanto del estado actual como de la entrada o entradas (*circuitos Mealy*).

REVISIÓN DE LA SECCIÓN 9.4

1. Un flip-flop J - K se encuentra actualmente en estado RESET y tiene que pasar al estado SET en el siguiente impulso de reloj. ¿Cuáles tienen que ser los valores de J y K?
2. Un flip-flop J-K se encuentra actualmente en estado SET y tiene que permanecer en dicho estado durante el siguiente impulso de reloj. ¿Cuáles tienen que ser los valores de J y K?
3. Un contador binario se encuentra en el estado $Q_3Q_2Q_1Q_0 = 1010$.
   (a) ¿Cuál es el estado siguiente?
   (b) ¿Qué condición tiene que existir en cada entrada de los flip-flops para asegurar que pasa al estado siguiente correcto con el impulso de reloj?

9.5 ■ CONTADORES EN CASCADA

Los contadores se pueden conectar en cascada para conseguir trabajar con módulos mayores. En esencia, conexión en cascada significa que la salida de la última etapa de un contador excita la entrada del siguiente contador.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Determinar el módulo global de los contadores en cascada. ■ Analizar el diagrama de tiempos de una configuración de contadores en cascada. ■ Utilizar contadores en cascada como divisores de frecuencia. ■ Utilizar contadores en cascada para conseguir secuencias específicas truncadas.

Un ejemplo de dos contadores conectados en **cascada** se muestra en la Figura 9.38, para el caso de dos contadores asíncronos de 2 y 3 bits. El cronograma se puede ver en la Figura 9.39.

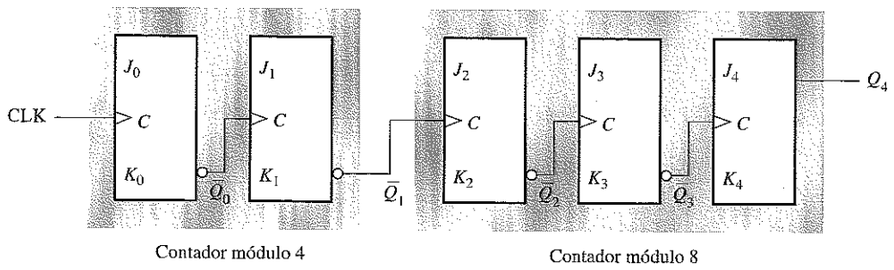


FIGURA 9.38
Dos contadores en cascada (todas las entradas J y K están a nivel ALTO).

Obsérvese que, en el cronograma de la Figura 9.39, la salida final del contador de módulo 8, Q_4, se produce una vez por cada 32 impulsos de reloj de entrada. El módulo global de los contadores en cascada es 32, es decir, actúan como un contador de división por 32. En general, el módulo global de los contadores en cascada es igual

al producto de los módulos individuales. Por ejemplo, para el contador de la Figura 9.38, el módulo global es 4 x 8 = 32.

NOTAS INFORMÁTICAS
El contador de marca temporal (TSC) mencionado en la anterior nota informática, es un contador de 64 bits. Es interesante observar que si a este contador (o cualquier contador de 64 bits de módulo completo) se le aplica una frecuencia de 100 MHz, tardará 5.849 años en pasar por todos sus estados y alcanzar su valor de cuenta terminal. Por el contrario, un contador de 32 bits de módulo completo recorrerá todos sus estados en aproximadamente 43 segundos cuando se aplique una señal de reloj de 100 MHz.

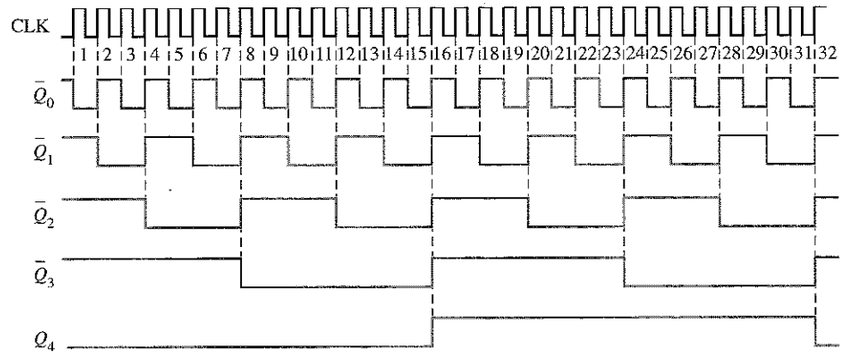


FIGURA 9.39
Cronograma para la configuración de contadores en cascada de la Figura 9.38.

Cuando se conectan contadores síncronos en una configuración en cascada, es necesario utilizar las funciones de habilitación de cuenta y de fin de cuenta para conseguir trabajar con módulos mayores. En algunos dispositivos, la habilitación de cuenta se denomina simplemente *CTEN* (count enable) o con cualquier otra designación, y la señal de fin de cuenta (*TC, terminal count)* es análoga a la salida de propagación de reloj (*RCO*) de algunos circuitos integrados contadores.

La Figura 9.40 muestra dos contadores de décadas conectados en cascada. El valor de fin de cuenta (*TC*) del contador 1 se conecta a la entrada de habilitación de cuenta (*CTEN*) del contador 2. El contador 2 se inhibe cuando su entrada *CTEN* está a nivel BAJO, hasta que el contador 1 alcanza su estado final y la salida del valor de fin de cuenta pasa a nivel ALTO. Este nivel ALTO activa ahora el contador 2, de modo que, cuando se produzca el primer impulso de reloj después de que el contador 1 alcance su valor de fin de cuenta (CLK10), el contador 2 pasa de su estado inicial a su segundo estado. Al terminar el segundo ciclo del contador 1 (cuando el contador 1 alcanza el valor de fin de cuenta por segunda vez), el contador 2 se encuentra de nuevo activado y avanza al estado siguiente. Esta secuencia se repite indefinidamente. Dado que se trata de contadores de décadas, el contador 1 tiene que pasar por diez ciclos completos antes de que el contador 2 complete su primer ciclo. En otras palabras, por cada diez ciclos del contador 1, el contador 2 realiza un único ciclo. Por tanto, el contador 2 completará un ciclo después de 100 impulsos de reloj. El módulo global de estos dos contadores en cascada es 10 x 10 = 100.

Si lo consideramos como un divisor de frecuencia, el circuito de la Figura 9.40 divide la frecuencia de entrada de reloj entre 100. Los contadores en cascada se utilizan a menudo para dividir una señal de reloj de alta frecuencia, y obtener impulsos de frecuencias precisas. Las configuraciones de los contadores en cascada utilizadas para estos propósitos se denominan algunas veces *cadenas de división.*

Por ejemplo, supongamos que tenemos una frecuencia de reloj básica de 1 MHz y que se desea obtener 100 kHz, 10 kHz y 1 kHz. Para ello, se puede utilizar una serie

de contadores de décadas en cascada. Si la señal de 1 MHz se divide entre 10, la salida tendrá una frecuencia de 100 kHz. Si dividimos después la señal de 100 kHz entre 10, la salida será una señal de 10 kHz. Otra división por 10 dará la señal de 1 kHz. La implementación de estas cadenas de división se muestra en la Figura 9.41.

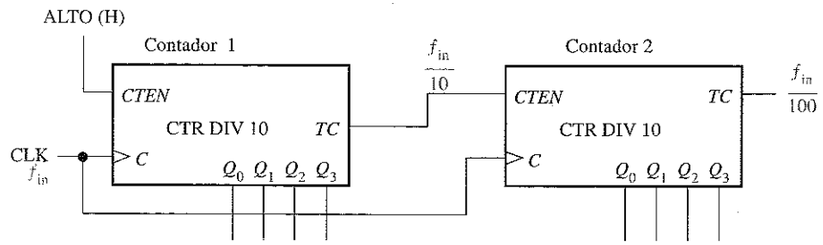


FIGURA 9.40
Contador de módulo 100, que utiliza dos contadores de décadas en cascada.

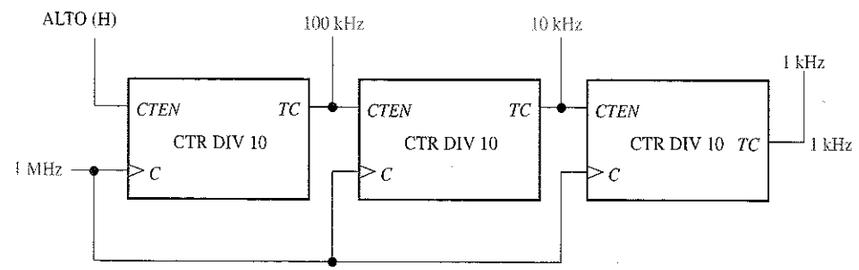


FIGURA 9.41
Tres contadores de décadas en cascada forman un divisor de frecuencia por 1000, con salidas intermedias de división por 100 y por 10.

EJEMPLO 9.7 Determinar el módulo global de las dos configuraciones de contadores en cascada de la Figura 9.42.

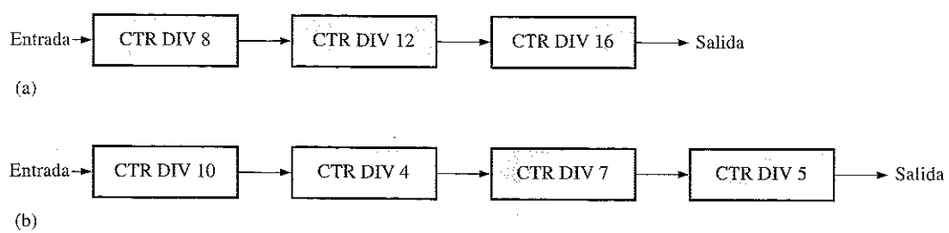


FIGURA 9.42

Solución. En la Figura 9.42(a), el módulo global para la configuración de 3 contadores es:

$$8 \times 12 \times 16 = \mathbf{1536}$$

En la Figura 9.42(b), el módulo global para la configuración de 4 contadores es:

$$10 \times 4 \times 7 \times 5 = \mathbf{1400}$$

Problema relacionado. ¿Cuántos contadores de décadas en cascada son necesarios para dividir una frecuencia de reloj entre 100.000?

EJEMPLO 9.8 Utilizar contadores 74HC160 para obtener una señal de 10 kHz a partir de una señal de reloj de 1 MHz. Dibujar el diagrama lógico.

Solución. Para obtener 10 kHz a partir de una señal de reloj de 1 MHz se requiere un factor de división de 100. Se tienen que conectar en cascada dos contadores 74HC160, como se muestra en la Figura 9.43. El contador de la izquierda produce un impulso *RCO* por cada 10 impulsos de reloj. El contador de la derecha produce un impulso *RCO* por cada 100 impulsos de reloj.

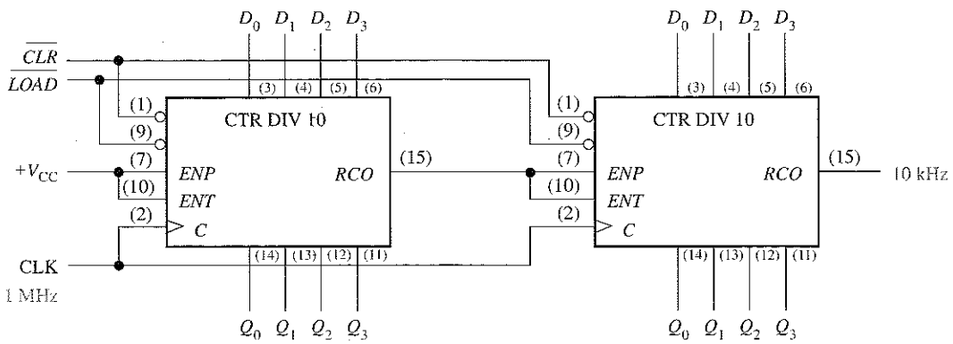


FIGURA 9.43
Contador divisor por 100 que utiliza dos contadores de décadas 74HC160.

Problema relacionado. Determinar la frecuencia de la onda de la salida Q_0 del segundo contador (el de la derecha) en la Figura 9.43.

Circuitos integrados contadores en cascada con secuencias truncadas

El estudio precedente ha mostrado cómo conseguir un módulo global (factor de división) que sea igual al producto de los módulos individuales de los contadores conectados en cascada. Esto se denomina *conexión en cascada de módulo completo*.

A menudo, una aplicación requiere un módulo global menor que el que se puede conseguir con la conexión en cascada de módulo completo. Es decir, se tiene que implementar una secuencia truncada con contadores en cascada. Para ilustrar este método, utilizaremos la configuración de contadores en cascada de la Figura 9.44. Este circuito particular utiliza cuatro contadores binarios de 4 bits 74HC161. Si estos cuatro contadores (dieciséis bits en total) fueran dispuestos mediante una conexión en cascada de módulo completo, el módulo sería:

$$2^{16} = 65.536$$

Vamos a suponer que una cierta aplicación requiere un contador-divisor por 40.000 (módulo 40.000). La diferencia entre 65.536 y 40.000 es 25.536, que es el número de estados que tienen que ser *eliminados* de la secuencia de módulo completo. La técnica utilizada en el circuito de la Figura 9.44 sirve para inicializar los contadores en cascada en el estado 25.536 (63C0 en hexadecimal) cada vez que se inicia un nuevo ciclo, de forma que cuenten desde 25.536 hasta 65.536 en cada ciclo completo. Por tanto, cada ciclo completo del contador tiene 40.000 estados.

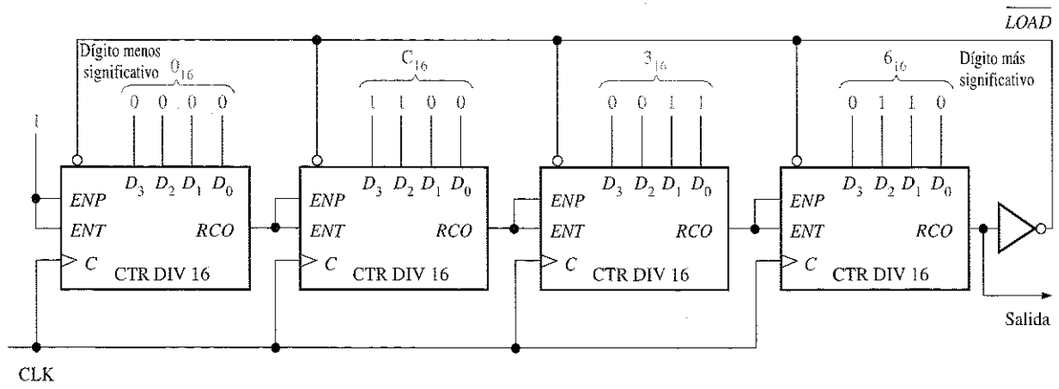


FIGURA 9.44
Contador divisor por 40.000 que utiliza contadores binarios de 4 bits 74HC161. Obsérvese que las entradas paralelas de datos se muestran en orden binario (el bit más a la derecha, D_0, es el LSB de cada contador).

Obsérvese en la Figura 9.44 que la salida *RCO* del contador más a la derecha está invertida y se aplica a la entrada $\overline{LOAD}$ de cada contador de 4 bits. Cada vez que el contador alcanza su valor de fin de cuenta de 65.536, *RCO* se pone a nivel ALTO y origina que el número que hay en sus entradas paralelas de datos ($63C0_{16}$) se cargue en el contador síncrono con el impulso de reloj. Por tanto, se produce un impulso *RCO* en el contador de 4 bits de más a la derecha por cada 40.000 impulsos de reloj.

Con esta técnica, se puede conseguir cualquier módulo, cargando el contador con el estado inicial apropiado en cada ciclo.

REVISIÓN DE LA SECCIÓN 9.5

1. ¿Cuántos contadores de décadas son necesarios para implementar un contador divisor por 1000 (módulo 1000)? ¿Y uno divisor por 10.000?
2. Mostrar mediante diagramas de bloques generales cómo conseguir cada uno de los siguientes dispositivos, utilizando un flip-flop, un contador de décadas, un contador binario de 4 bits o cualquier combinación de éstos:
   (a) Contador divisor por 20 **(b)** Contador divisor por 32
   (c) Contador divisor por 160 **(d)** Contador divisor por 320

9.6 ■ DECODIFICACIÓN DE CONTADORES

En muchas aplicaciones, es necesario decodificar algunos o todos los estados del contador. La decodificación de un contador implica la utilización de decodificadores o de puertas lógicas para determinar cuándo se encuentra el contador en un determinado estado binario de su secuencia. Por ejemplo, la función de fin de cuenta estudiada previamente es una decodificación de un único estado (el último estado) de la secuencia del contador.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Implementar la lógica de decodificación para cualquier estado de la secuencia de un contador. ■ Explicar por qué aparecen glitches en la lógica de decodificación de un contador. ■ Utilizar el método de validación (strobing) para eliminar los glitches en la decodificación.

Supongamos que se desea decodificar el estado binario 6 (110) de un contador binario de 3 bits. Esto se puede realizar como se muestra en la Figura 9.45. Cuando $Q_2 = 1$, $Q_1 = 1$ y $Q_0 = 0$, aparece un nivel ALTO en la salida de la puerta de decodificación, indicando que el contador se encuentra en el estado 6. Esto se denomina *decodificación activa a nivel ALTO*. Reemplazando la puerta AND por una puerta NAND obtenemos una decodificación activa a nivel BAJO.

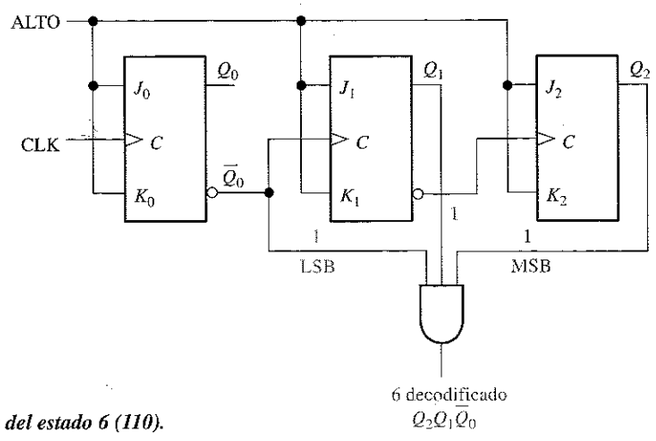


FIGURA 9.45
Decodificación del estado 6 (110).

EJEMPLO 9.9 Implementar la decodificación de los estados binarios 2 y 7 de un contador síncrono de 3 bits. Dibujar el diagrama de tiempos completo y las formas de onda de salida de las puertas de decodificación. 2 binario = $\overline{Q}_2Q_1\overline{Q}_0$ y 7 binario = $Q_2Q_1Q_0$.

Solución. Véase la Figura 9.46. El contador de 3 bits fue explicado anteriormente en la Sección 9.2 (Figura 9.14).

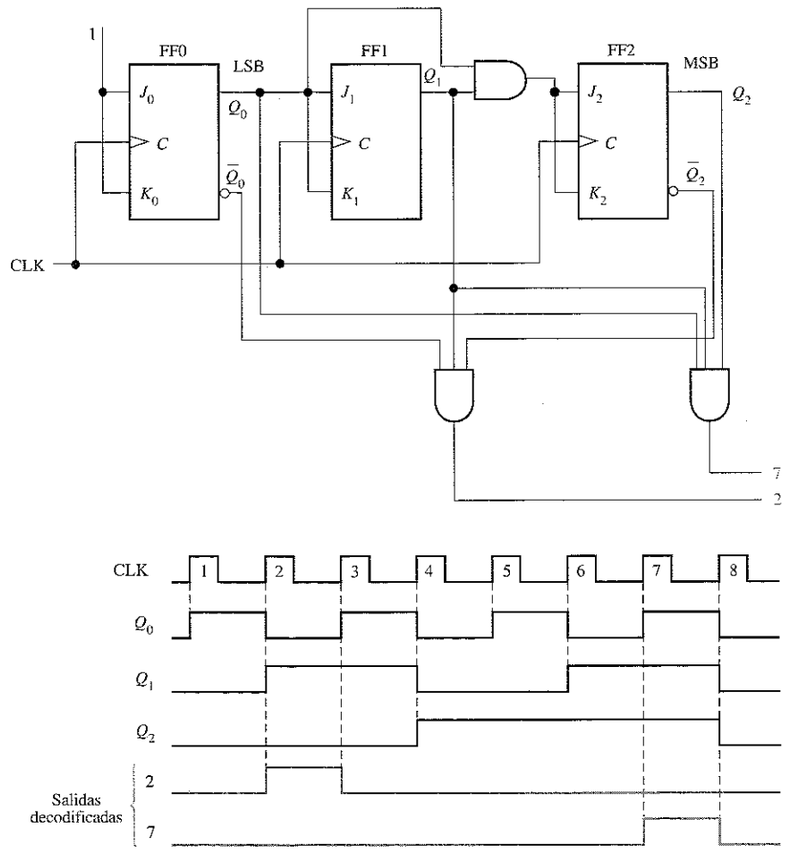


FIGURA 9.46
Contador de 3 bits con decodificación activa a nivel ALTO de los estados 2 y 7.

Problema relacionado. Realizar la lógica de decodificación del estado 5 de un contador de 3 bits.

Glitches en la decodificación

En el Capítulo 6 se introdujo el problema de los glitches producidos por el proceso de decodificación. Como ya se ha visto, los retardos de propagación debidos al efecto del

retraso en los contadores asíncronos origina estados transitorios, en los que las salidas del contador están variando en instantes de tiempo ligeramente distintos. Estos estados transitorios producen picos de tensión de corta duración (glitches) no deseados, que aparecen en las salidas del decodificador conectado al contador. El problema de los glitches puede también aparecer en cierta medida en los contadores síncronos, ya que los retardos de propagación entre el reloj y las salidas Q de cada flip-flop del contador pueden diferir ligeramente.

FIGURA 9.47
Contador básico de décadas (BCD) y decodificador.

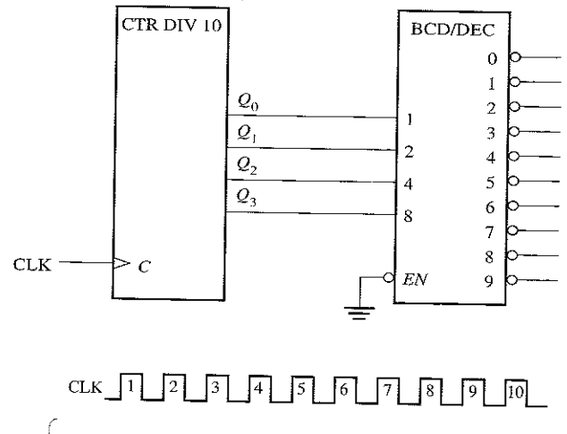
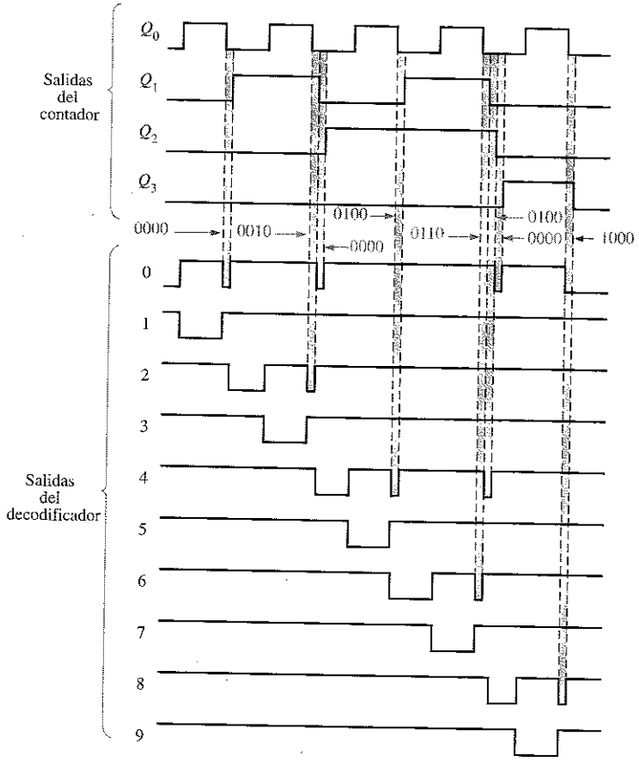


FIGURA 9.48
Salidas con glitches procedentes del decodificador de la Figura 9.47. Las anchuras de los glitches están exageradas y, generalmente, tienen un valor de unos pocos nanosegundos.



La Figura 9.47 muestra un contador asíncrono básico de décadas BCD conectado a un decodificador BCD-decimal. Para ver qué es lo que ocurre, vamos a examinar el diagrama de tiempos de la Figura 9.48, en el que se tienen en cuenta los retardos de propagación. Obsérvese que estos retardos originan estados erróneos de corta duración. El valor del estado binario falso en cada transición crítica se indica en el diagrama. Los glitches resultantes pueden verse en las salidas del decodificador.

Una manera de eliminar los glitches consiste en activar las salidas del decodificador después de que los impulsos de ruido hayan tenido tiempo de desaparecer. Este método se conoce como *validación (strobing)* y se puede implementar, en este caso, como se muestra en la Figura 9.49, utilizando el nivel BAJO del reloj para activar el decodificador. El diagrama de tiempos mejorado que se obtiene se presenta en la Figura 9.50.

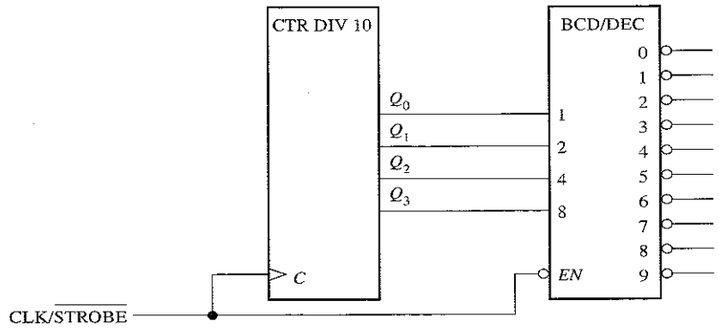


FIGURA 9.49
Contador asíncrono básico de décadas y decodificador con validación (strobe) para eliminar los impulsos de ruido.

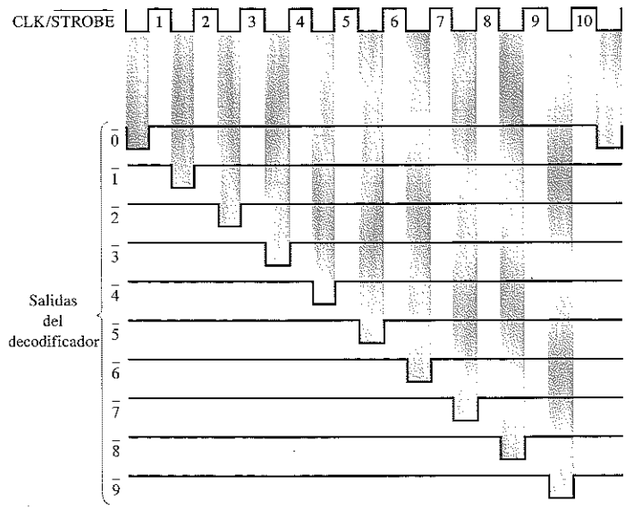


FIGURA 9.50
Salidas del decodificador con validación para el circuito de la Figura 9.49.

REVISIÓN DE LA SECCIÓN 9.6

1. ¿Cuáles son los posibles estados transitorios cuando un contador binario de 4 bits asíncrono cambia del
   (a) estado 2 al 3? **(b)** estado 3 al 4?
   (c) estado 10_{10} al 11_{10}? **(d)** estado 15 al 0?

9.7 ■ APLICACIONES DE LOS CONTADORES

Los contadores digitales son dispositivos muy versátiles y útiles que se pueden encontrar en muchas aplicaciones. En esta sección, se presentan varias aplicaciones representativas de los contadores.

Al finalizar esta sección, el lector deberá ser capaz de:

- Describir cómo se utilizan los contadores en un sistema básico de reloj digital.
- Explicar cómo se implementa un contador-divisor por 60 y cómo se utiliza en un reloj digital. ■ Explicar cómo se implementan los contadores de horas. ■ Estudiar cómo aplicar un contador en un sistema de control de un aparcamiento de automóviles.
- Describir cómo se utiliza un contador en un proceso de conversión de datos paralelo-serie

Ejemplos de aplicación

El reloj digital. Un ejemplo típico de aplicación de los contadores son los sistemas de control de tiempo. La Figura 9.51 es un diagrama lógico simplificado de un reloj digital, que presenta en el display segundos, minutos y horas. En primer lugar, se transforma una tensión alterna sinusoidal de 60 Hz en un tren de impulsos a 60 Hz y, posteriormente, se divide para obtener un tren de impulsos a 1 Hz, mediante un contador-divisor por 60, formado por un divisor por 10 seguido de un un divisor por 6. La cuenta de *minutos* y de *segundos* se genera también mediante contadores-divisores por 60, operación que se puede ver en detalle en la Figura 9.52. Estos contadores cuentan desde 0 hasta 59 y luego vuelven al estado 0; en esta implementación particular se utilizan contadores de décadas síncronos 74LS160A. Obsérvese que la etapa del divisor por 6 está constituida por un contador de décadas con una secuencia truncada, que se logra utilizando el estado 6 decodificado para borrar, en modo asíncrono, el contador. El valor de final de cuenta 59 también se codifica para activar el siguiente contador de la cadena.

El contador de *horas* se implementa mediante un contador de décadas y un flip-flop, tal y como se muestra en la Figura 9.53. Supongamos que, inicialmente, tanto el contador de décadas como el flip-flop se encuentran en estado RESET, y que la puerta de decodificación del estado 12 está en nivel ALTO. El contador de décadas avanza pasando por todos sus estados desde cero hasta nueve y, al pasar de nueve a cero para iniciar un nuevo ciclo, el flip-flop bascula al estado SET ($J = 1$, $K = 0$). Esto hace que se ilumine un 1 en el display, que indica el dígito de las decenas de horas. El valor total de cuenta es ahora 10 (el contador de décadas está en estado cero y el flip-flop en estado SET).

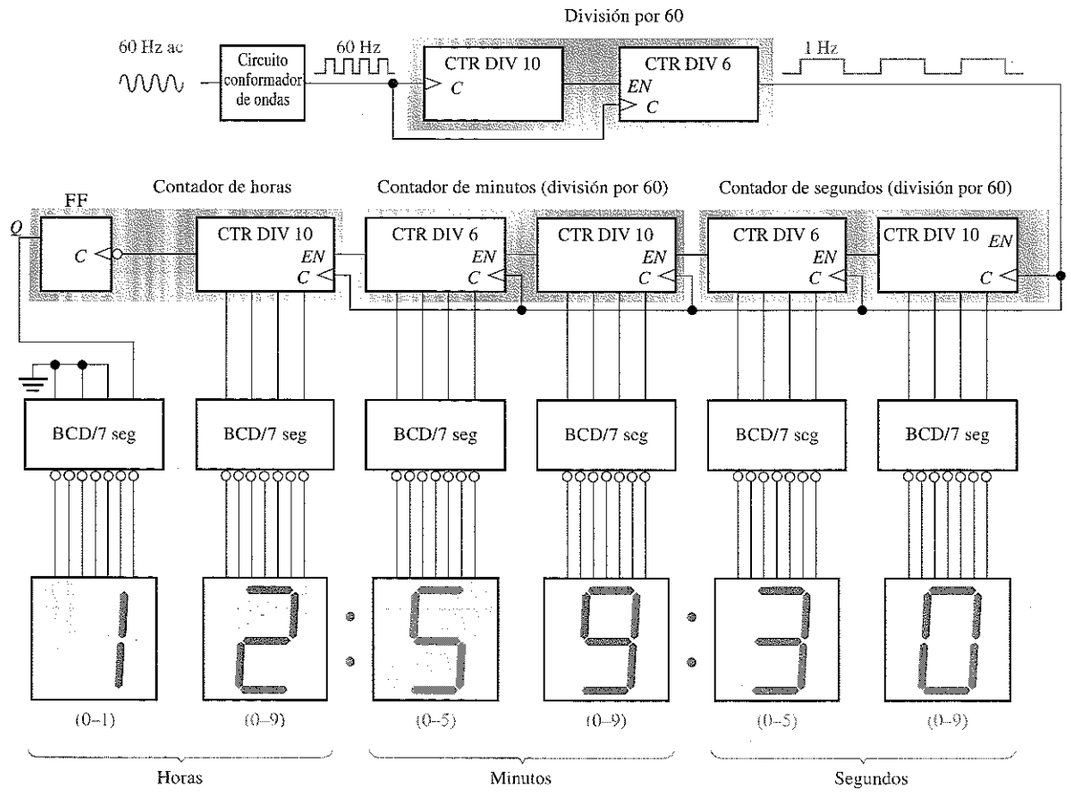


FIGURA 9.51
Diagrama lógico simplificado de un reloj digital de 12 horas. Los detalles del circuito lógico, con sus dispositivos específicos, se muestran en las Figuras 9.52 y 9.53.

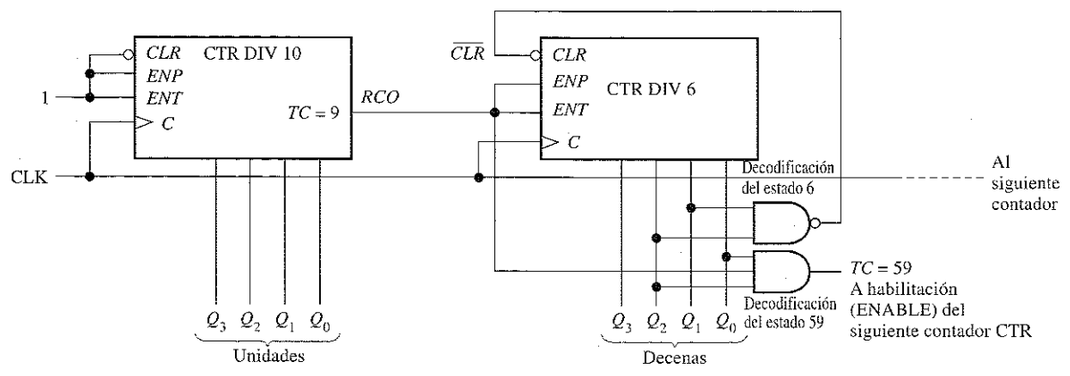


FIGURA 9.52
Diagrama lógico de un contador-divisor por 60 que utiliza contadores de décadas síncronos 74LS160A. Obsérvese que las salidas están en orden binario (el bit más a la derecha es el menos significativo).

NOTAS INFORMÁTICAS
Las computadoras disponen de un contador interno que puede programarse para distintas frecuencias y duraciones de tonos, produciendo "música". Para seleccionar un tono concreto, la instrucción programada selecciona un valor divisor que es enviado al contador. El divisor configura al contador de modo que divida la frecuencia básica del reloj del periférico de 1,19 MHz (valor estándar para todos los PC), para generar un tono de audio. La duración de un tono también se puede definir mediante una instrucción de programa; por tanto, se utiliza un contador básico para generar melodías controlando la frecuencia y duración de los tonos.

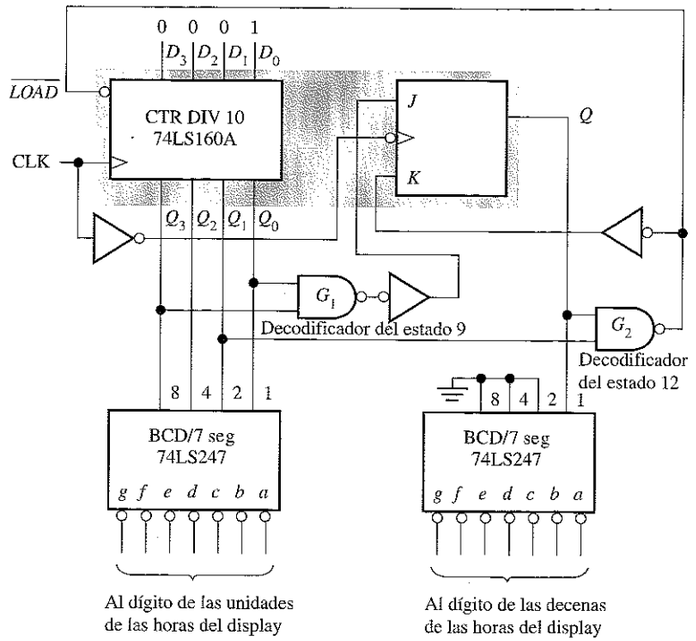


FIGURA 9.53
Diagrama lógico del contador y de los decodificadores de horas. Obsérvese que, en las entradas y salidas del contador, el bit más a la derecha es el LSB.

A continuación, el número total avanza hasta once y luego a doce. En el estado 12, la salida Q_2 del contador de décadas es un nivel ALTO, el flip-flop sigue en estado SET y, por tanto, la salida de la puerta de decodificación está a nivel BAJO. Esto activa la entrada $\overline{LOAD}$ del contador de décadas. En el siguiente impulso de reloj, el contador de décadas es inicializado en el estado 1 a través de las entradas de datos, y el flip-flop pasa al estado de RESET ($J = 0$, $K = 1$). Como puede verse, esta lógica hace que siempre el contador inicie un nuevo ciclo pasando de doce a uno, en lugar de a cero.

Sistema de control de un aparcamiento de automóviles. Ahora vamos a ver una sencilla aplicación que ilustra cómo puede resolver un contador ascendente/descendente un problema cotidiano. El problema consiste en concebir una forma de control de las plazas disponibles en un aparcamiento de 100 plazas y, en caso de que esté lleno, hacer que se encienda una luz de aviso y que se baje una barrera a la entrada.

Un sistema que resuelve este problema está constituido por: (1) sensores optoelectrónicos en la entrada y salida del aparcamiento; (2) un contador ascendente/descendente y su circuitería asociada, y (3) un circuito de interfaz que utilice la salida del contador para encender o apagar la luz de COMPLETO, así como para subir o bajar la barrera de entrada. En la Figura 9.54 se presenta un diagrama general de bloques de este sistema.

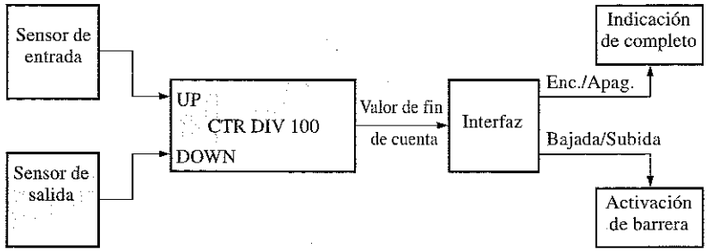


FIGURA 9.54
Diagrama de bloques funcional para el control del aparcamiento.

En la Figura 9.55 se muestra un diagrama lógico del contador ascendente/descendente. Está formado por dos contadores de décadas ascendentes/descendentes 74LS190 conectados en cascada. Su funcionamiento se describe en los siguientes párrafos.

El contador se inicializa previamente en el estado 0 mediante las entradas de datos paralelo, las cuales no se muestran. Cada automóvil que entra en el aparcamiento interrumpe la trayectoria de un haz de luz, activando un sensor que produce un impulso eléctrico. Este impulso positivo activa el latch S-R con su flanco anterior. El nivel BAJO en la salida $\overline{Q}$ del latch hace que el contador entre en el modo de trabajo ascendente. Además, el impulso producido por el sensor pasa a través de la puerta NOR y aplica la señal de reloj al contador durante la transición de nivel BAJO a nivel ALTO de su flanco posterior. Cada vez que entra un coche en el aparcamiento, el contador avanza una posición (**se incrementa**). Cuando han entrado cien automóviles, el contador llega a su estado final (100_{10}). La salida *MAX/MIN* se pone a nivel ALTO y activa el circuito de interfaz (en el cual no entraremos), que enciende la luz de COMPLETO y baja la barrera para evitar que sigan entrando coches.

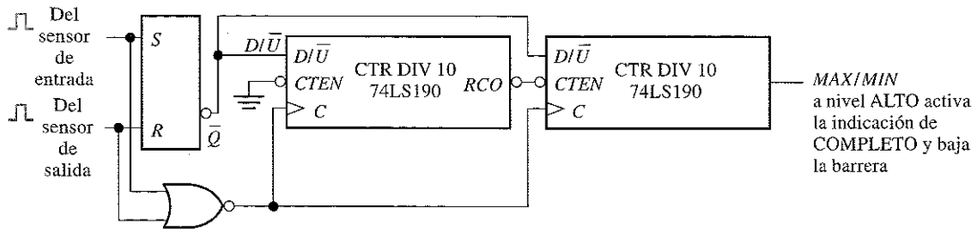


FIGURA 9.55
Diagrama lógico de un contador ascendente/descendente de módulo 100 para el control del aparcamiento.

Cuando sale un automóvil, un sensor optoelectrónico produce un impulso positivo, que pone a cero (RESET) el latch S-R y hace que el contador entre en modo descendente. El flanco posterior del reloj decrementa en una unidad el número que marque el contador. Si el aparcamiento está completo y sale un automóvil, la salida

MAX/MIN del contador pasa a nivel BAJO, haciendo que desaparezca la luz de COMPLETO y subiendo la barrera.

Conversión de datos paralelo-serie (multiplexación). En el Capítulo 6 ya expusimos un ejemplo simplificado de transmisión de datos utilizando técnicas de multiplexación y demultiplexación. Esencialmente, los bits de datos paralelos en las entradas del multiplexor se convierten en bits de datos en serie que se transmiten por una única línea. Se denominan *datos en paralelo* a un grupo de bits que se presentan simultáneamente sobre varias líneas paralelas. Se denomina *datos en serie* a un grupo de bits que se presentan secuencialmente por una única línea.

Normalmente, la conversión paralelo-serie se realiza utilizando un contador que proporcione una secuencia binaria para las entradas de selección de datos de un multiplexor/selector de datos, como muestra la Figura 9.56. Las salidas Q del contador de módulo 8 se conectan a las entradas de selección de datos de un multiplexor de 8 bits.

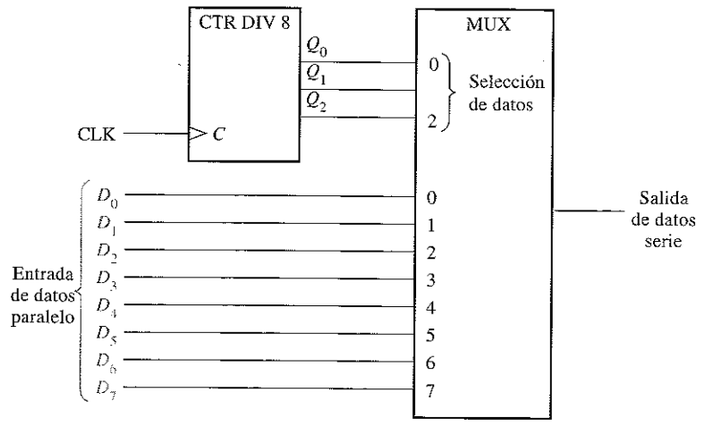


FIGURA 9.56
Lógica de conversión de datos paralelo-serie.

La Figura 9.57 es un diagrama de tiempos que muestra el funcionamiento de este circuito. El primer byte (grupo de ocho bits) de datos en paralelo se aplica a las entradas del multiplexor. A medida que el contador avanza a lo largo de su secuencia binaria desde cero hasta siete, cada bit, comenzando por D_0, se selecciona secuencialmente y pasa a través del multiplexor hasta la línea de salida. Después de ocho impulsos de reloj, el byte de datos ha sido convertido a formato serie y enviado a través de la línea de transmisión. Cuando el contador retorna a 0, el siguiente byte se aplica a las entradas de datos y se convierte secuencialmente en formato serie a medida que el contador pasa por sus ocho estados. Este proceso continúa repetidamente para convertir cada byte paralelo a serie. Uno de los problemas del final del capítulo (problema 52) requiere la implementación de este circuito con dispositivos específicos.

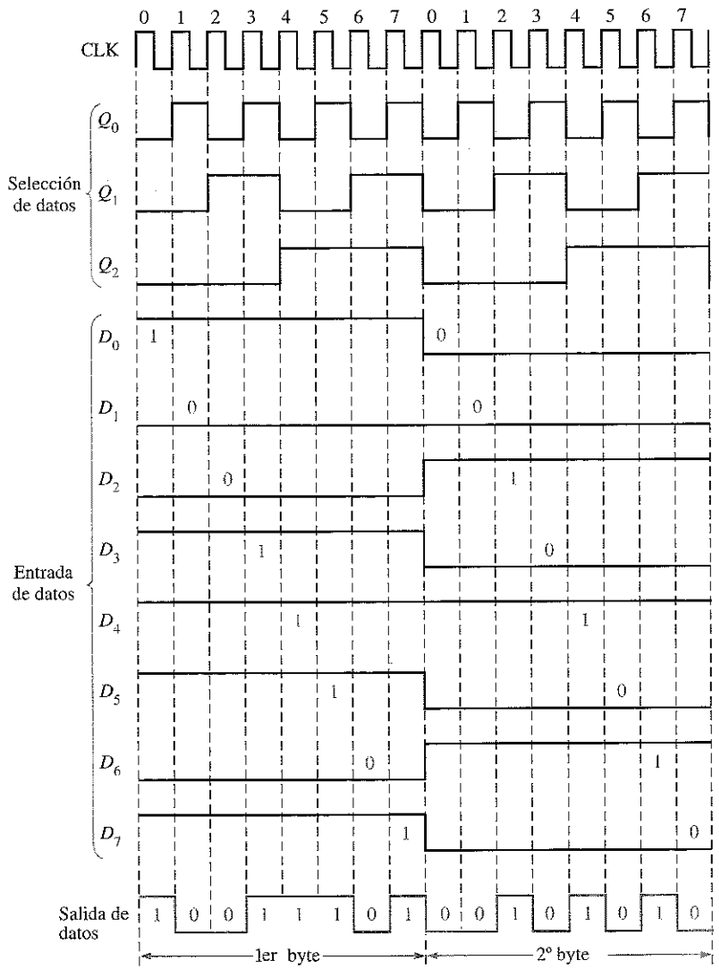


FIGURA 9.57
Diagrama de tiempos de la conversión paralelo-serie del circuito de la Figura 9.56.

REVISIÓN DE LA SECCIÓN 9.7

1. Explicar para qué sirve cada puerta NAND de la Figura 9.53.
2. Identificar las dos condiciones para iniciar un nuevo ciclo en el contador de horas de la Figura 9.51, y explicar para qué sirven.

9.8 ■ LOCALIZACIÓN DE AVERÍAS

La localización de averías en los contadores puede ser simple o muy complicada, dependiendo del tipo de contador y del tipo de error. Esta sección nos va a proporcionar un poco de práctica en la resolución de problemas en los circuitos secuenciales.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Detectar fallos en un contador. ■ Aislar los fallos en los contadores en cascada de módulo máximo. ■ Aislar los fallos en los contadores en cascada con secuencia truncada. ■ Determinar los fallos en los contadores implementados con flip-flops individuales.

Circuitos integrados contadores

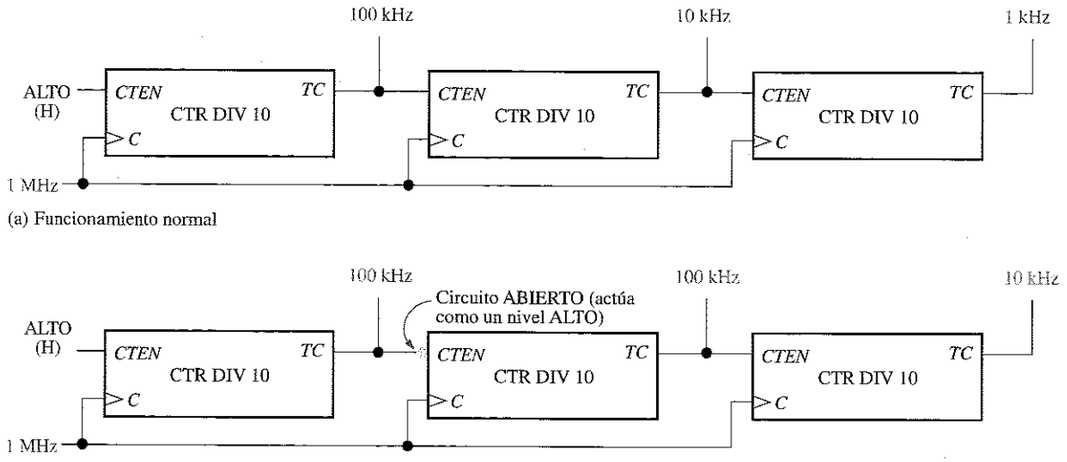
En un CI contador que tenga una secuencia que no esté controlada por alguna lógica externa, la única cosa que se puede comprobar (aparte de V_{CC} y tierra) es la posibilidad de tener entradas o salidas en circuito abierto o cortocircuitadas. Un CI contador casi nunca altera su secuencia de estados debido a un fallo interno, por lo que sólo se debe comprobar la actividad de los impulsos en las salidas Q, para detectar la existencia de circuitos abiertos o cortocircuitos. La ausencia de actividad de impulsos en una de las salidas Q indica que hay un circuito abierto o un cortocircuito interno. La ausencia de actividad de impulsos en todas las salidas Q indica que la entrada de reloj está fallando o que la entrada de borrado se mantiene en su estado activo.

Para comprobar la entrada de borrado, se aplica un nivel activo constante a la misma, a la vez que la señal de reloj. Si se obtiene un nivel BAJO en cada una de las salidas Q, el funcionamiento es correcto.

La capacidad de carga paralelo de un contador se puede comprobar activando la entrada $\overline{LOAD}$ y probando cada uno de los estados del siguiente modo: se aplican niveles BAJOS a las entradas de datos en paralelo, impulsos en la entrada de reloj y se comprueba que haya niveles BAJOS en todas las salidas Q. A continuación, se aplican niveles ALTOS a las entradas de datos en paralelo, se introducen impulsos en la entrada de reloj y se comprueba que haya niveles ALTOS en todas las salidas Q.

Circuitos integrados contadores en cascada con módulo máximo

Un fallo en uno de los contadores de una cadena de contadores en cascada puede afectar a todos los demás contadores que le siguen. Por ejemplo, si la entrada de habilitación de cuenta de un contador está en circuito abierto, actúa como si fuera un nivel ALTO (TTL) y el contador estará siempre activado. Este tipo de fallo en uno de los contadores hará que ese contador funcione a la máxima velocidad de reloj y también que todos los demás contadores que lo sigan funcionen a velocidades mayores de las normales. Esto se ilustra en la Figura 9.58 para un contador-divisor por 1000 en cascada, donde una entrada de habilitación de cuenta *(CTEN)* en circuito abierto actúa como un nivel ALTO TTL y activa continuamente al segundo contador. Otro de los fallos que pueden afectar a las "etapas secundarias" de los contadores pueden ser entradas de reloj o salidas de valor de fin de cuenta en circuito abierto o cortocircuitadas. En algunas de estas situaciones, se puede observar actividad de impulsos, pero a frecuencias erróneas. En este caso se tendrán que realizar medidas de frecuencia exactas.



(a) Funcionamiento normal

(b) La entrada de habilitación de cuenta (*CTEN*) del segundo contador está en circuito abierto.

FIGURA 9.58
Ejemplo de fallo que afecta a los sucesivos contadores en una configuración en cascada.

Contadores en cascada con secuencias truncadas

La secuencia de números de un contador en cascada con una secuencia truncada, como el de la Figura 9.59, puede dar lugar a otros tipos de fallos, además de los mencionados para los contadores en cascada de módulo máximo. Por ejemplo, un fallo en una de las entradas de datos en paralelo, la entrada $\overline{LOAD}$ o el inversor pueden alterar el valor de inicialización y, por tanto, cambiar el módulo del contador.

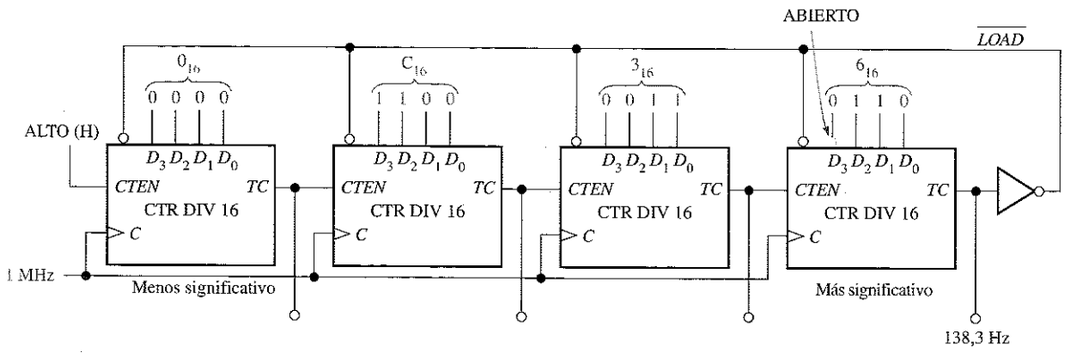


FIGURA 9.59
Ejemplo de fallo en un contador en cascada con secuencia truncada.

Por ejemplo, supongamos que la entrada D_3 del contador más significativo de la Figura 9.59 está en circuito abierto y actúa como un nivel ALTO. Entonces, en lugar

de hacer la reinicialización en el estado 6_{16} (0110), se hará en el estado E_{16} (1110). De esta manera, en lugar de comenzar en $63C0_{16}$ (25.536_{10}), cada vez que el contador inicie un nuevo ciclo, la secuencia comenzará en $E3C0_{16}$ (58.304_{10}). Esto hace que varíe el módulo del contador de 40.000 a 65.536 - 58.304 = 7.232.

Para comprobar este contador se aplica una frecuencia de reloj conocida, por ejemplo 1 MHz, y se mide la frecuencia de salida en el terminal del valor de fin de cuenta. Si el contador está funcionando adecuadamente, la frecuencia de salida será:

$$f_{out} = \frac{f_{in}}{\text{módulo}} = \frac{1\text{MHz}}{40.000} = 25 \text{ Hz}$$

En este caso, el fallo específico descrito en el párrafo anterior hará que la frecuencia de salida sea:

$$f_{out} = \frac{f_{in}}{\text{módulo}} = \frac{1\text{MHz}}{7.232} = 138{,}3 \text{ Hz}$$

EJEMPLO 9.10 Se realizan medidas de frecuencia en el contador truncado de la Figura 9.60 tal y como se indica. Determinar si el contador está funcionando adecuadamente y, si no es así, determinar cuál es el fallo.

Solución. Se comprueba que la frecuencia medida en TC 4 es la correcta. Si así es, el contador está funcionando perfectamente.

$$\begin{aligned}\text{módulo truncado} &= \text{módulo completo} - \text{valor de ajuste previo} \\ &= 16^4 - 82C0_{16} \\ &= 65.536 - 33.472 = 32.064\end{aligned}$$

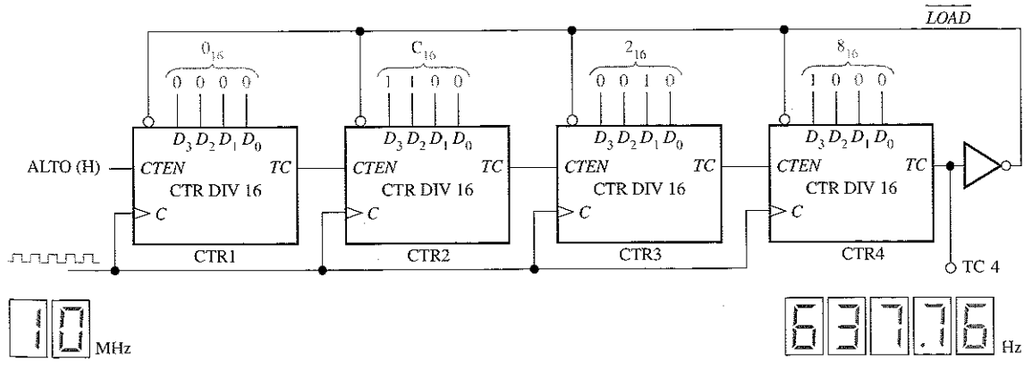


FIGURA 9.60

La frecuencia correcta en TC 4 es

$$f_4 = \frac{10 \text{ MHz}}{32.064} = 311{,}88 \text{ Hz}$$

Sin embargo, al medir, detectamos que hay un problema. La frecuencia medida de 637,76 Hz no concuerda con la frecuencia correcta calculada de 311,88 Hz.

Para comprobar el contador que falla, se determina el módulo truncado real del siguiente modo:

$$\text{módulo} = \frac{f_{out}}{f_{in}} = \frac{10\text{MHz}}{637{,}76\text{ Hz}} = 15.680$$

Debido a que el módulo truncado tiene que ser 32.064, lo más probable es que la inicialización del contador se esté haciendo con un valor erróneo cuando se inicia un nuevo ciclo. El valor de inicialización real se determina de la manera siguiente:

$$\begin{aligned}\text{módulo truncado} &= \text{módulo completo} - \text{valor de inicialización}\\ \text{valor de inicialización} &= \text{módulo completo} - \text{módulo truncado}\\ &= 65.536 - 15.680\\ &= 49.856\\ &= \text{C2C0}_{16}\end{aligned}$$

Esto muestra que el valor de inicialización del contador es, cada vez que se inicia un nuevo ciclo, $C2C0_{16}$ en lugar de $82C0_{16}$

La carga de los contadores 1, 2 y 3 se realiza correctamente, pero no la del contador 4. Dado que $C_{16} = 1100_2$, la entrada D_2 del contador 4 está a nivel ALTO cuando debería estar a nivel BAJO. Lo más probable es que este fallo esté causado por una entrada en **circuito abierto**. Hay que comprobar también un circuito abierto externo causado por una mala soldadura en las conexiones, un conductor roto o un pin curvado del circuito integrado. Si no se detecta ninguno de estos fallos, debe reemplazarse el CI, y el contador funcionará correctamente.

Problema relacionado. Determinar cuál sería la frecuencia de salida en TC 4 si la entrada D_3 del contador 3 estuviera en circuito abierto.

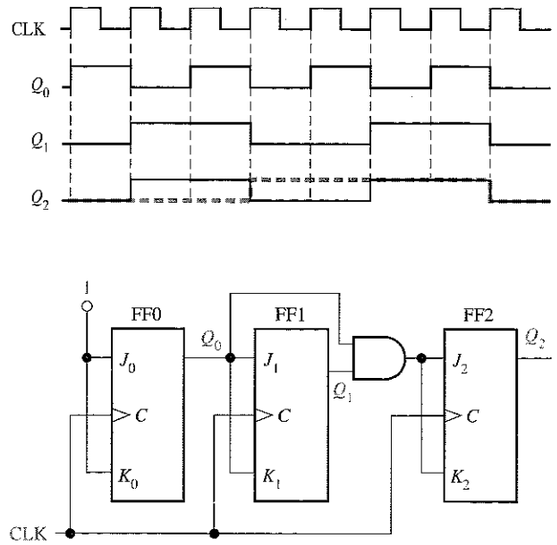
Contadores implementados con flip-flops individuales

Los contadores implementados con flip-flops individuales y circuitos integrados de puertas son, algunas veces, más difíciles de comprobar en caso de fallo, ya que hay muchas más entradas y salidas con conexiones externas que en un CI contador. La secuencia de un contador se puede alterar por la existencia de un único circuito abierto o un cortocircuito en una entrada o salida, como nos muestra el Ejemplo 9.11.

EJEMPLO 9.11 Supongamos que se observan las formas de onda de salida que se indican para el contador de la Figura 9.61. Determinar si existe algún problema en el contador.

Solución. La forma de onda Q_2 es incorrecta. La forma de onda correcta se indica mediante una línea discontinua. Puede observarse que la forma de onda Q_2 tiene exactamente la misma forma que Q_1. Esto indica que la misma señal que está haciendo bascular a FF1 controla también a FF2.

FIGURA 9.61



Si comprobamos las entradas J y K de FF2, encontramos una señal que tiene la misma forma que Q_0. Este resultado indica que Q_0 pasa de alguna manera a través de la puerta AND. Esto sólo puede ocurrir si la entrada Q_1 de la puerta AND está siempre a nivel ALTO. Pero acabamos de ver que Q_1 tiene una forma de onda correcta. Esta observación nos conduce a la conclusión de que la entrada inferior de la puerta AND tiene que estar, internamente, en circuito abierto, por lo que actúa como un nivel ALTO. Es necesario entonces reemplazar la puerta AND y volver a comprobar el circuito.

Problema relacionado. Describir la salida Q_2 del contador de la Figura 9.61 si la salida Q_1 de FF1 está en circuito abierto.

Consejos Prácticos

Cuando se observa la relación temporal entre dos señales digitales en un osciloscopio de doble traza, la forma adecuada de disparar el osciloscopio es mediante la señal más lenta de las dos. La razón de esto es que la señal más lenta dispone de menos puntos de disparo que la señal más rápida, por lo que no existirá ambigüedad en el inicio del barrido. El disparo en modo vertical utiliza una señal compuesta de ambos canales y nunca se debería emplear para determinar información temporal absoluta. Dado que, generalmente, las señales de reloj son las más rápidas en un sistema digital, no se deberían utilizar para el disparo.

REVISIÓN DE LA SECCIÓN 9.8

1. ¿Cuáles son los fallos que pueden causar que el contador de la Figura 9.58 no tenga actividad de impulsos en ninguna de las salidas *TC*?
2. ¿Qué ocurriría si el inversor de la Figura 9.60 tuviera una salida en circuito abierto?

9.9 ■ SÍMBOLOS LÓGICOS CON NOTACIÓN DE DEPENDENCIA*

Hasta este momento, se han introducido de una manera limitada los símbolos lógicos con notación de dependencia especificados en las normas ANSI/IEEE Standard 91-1984. En muchos casos, los nuevos símbolos no se apartan significativamente de los tradicionales. Sin embargo, existen diferencias considerables en algunos dispositivos, incluidos los contadores y otros dispositivos más complejos, con respecto a lo que estamos acostumbrados a ver. Aunque vamos a continuar utilizando principalmente los símbolos más tradicionales y familiares a lo largo del libro, se proporciona una breve descripción de los símbolos lógicos con notación de dependencia.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Interpretar los símbolos lógicos que incluyen notación de dependencia.
■ Interpretar el bloque común y los elementos individuales del símbolo de un contador. ■ Interpretar el símbolo de cualificación. ■ Explicar la dependencia de control. ■ Explicar la dependencia de modo. ■ Explicar la dependencia AND.

La notación de dependencia es fundamental en el estándar ANSI/IEEE. La notación de dependencia se utiliza junto con los símbolos lógicos para especificar las relaciones entre entradas y salidas, de forma que el funcionamiento lógico de un dispositivo específico pueda ser determinado enteramente a partir de su símbolo lógico, sin ningún conocimiento a priori de los detalles de su estructura interna y sin necesidad de ningún diagrama lógico detallado como referencia.

Se utiliza el contador binario síncrono de 4 bits 74HC163 como ejemplo. Para poder comparar, la Figura 9.62 muestra el símbolo de bloque tradicional y el símbolo ANSI/IEEE con notación de dependencia. La descripción básica del símbolo y de la notación de dependencia es la siguiente:

Bloque de control común. El bloque superior con las esquinas cortadas de la Figura 9.62(b) tiene varias entradas y una salida que se consideran comunes a todos los elementos del dispositivo y no son exclusivas de ningún elemento.

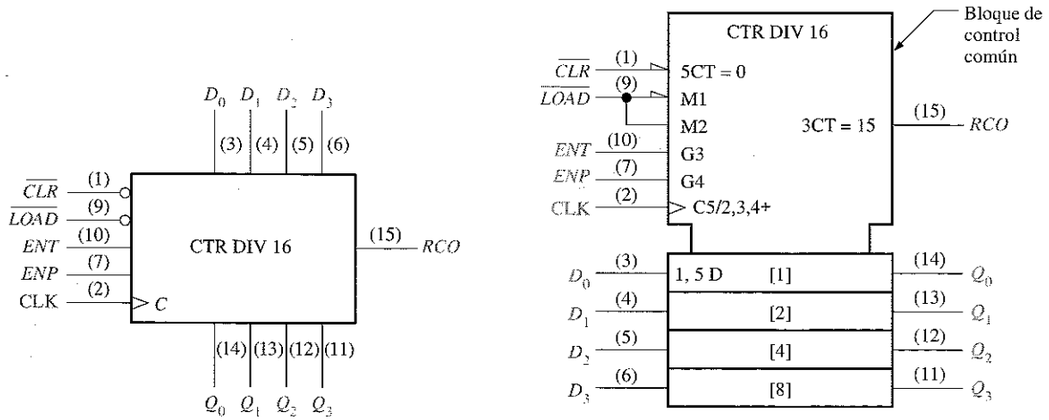
Elementos individuales. El bloque inferior de la Figura 9.62(b), que está dividido en cuatro secciones colindantes, representa los cuatro elementos de almacenamiento (flip-flops D) del contador, cuyas entradas son D_0, D_1, D_2 y D_3, y cuyas salidas son Q_0, Q_1, Q_2,y Q_3.

Símbolo de cualificación. La etiqueta "CTR DIV 16" de la Figura 9.62(b) identifica el dispositivo como un contador (CTR) con dieciséis estados (DIV 16).

Dependencia de control (C). Como se muestra en la Figura 9.62(b), la letra *C* denota dependencia del control. Las entradas de control normalmente activan o desactivan las entradas de datos (*D, J, K, S* y *R*) de un elemento de almacenamiento. La entrada *C* suele ser la entrada de reloj. En este caso, el dígito 5 que sigue a *C* (*C*5/2,3,4+) indica que las entradas etiquetadas con un prefijo 5 dependen del reloj (están sincronizadas con el reloj). Por ejemplo, 5CT = 0 en la entrada $\overline{CLR}$ indica que la función de borrado depende del reloj, es decir, es una entrada de borrado síncrona.

* Esta sección es opcional y su omisión no afectará al resto de los temas del libro.

Cuando la entrada $\overline{CLR}$ está a nivel BAJO (0), el contador se pone a cero ($CT = 0$) en el flanco de disparo del impulso de reloj. Además, la etiqueta 5D en la entrada del elemento de almacenamiento [1] indica que el almacenamiento de datos depende del reloj (está sincronizado con el reloj). Todas las etiquetas en el elemento de almacenamiento [1] se aplican también a los elementos [2], [4] y [8] que están situados por debajo, ya que no se indica en ellos una notación diferente.



(a) Símbolo tradicional

(b) Símbolo lógico estándar ANSI/IEEE 91-1984

FIGURA 9.62
El contador síncrono de 4 bits 74HC163.

Dependencia de modo (M). Como se muestra en la Figura 9.62(b), la letra *M* indica dependencia de modo. Se utiliza para indicar cómo dependen las funciones correspondientes a varias entradas o salidas del modo en que está funcionando el dispositivo. En este caso, el dispositivo tiene dos modos de funcionamiento. Cuando la entrada $\overline{LOAD}$ está a nivel BAJO (0), como se indica mediante el triángulo de entrada, el contador está en modo de carga (M1) en el que los datos de entrada (D_0, D_1, D_2 y D_3) se introducen de manera síncrona en los cuatro flip-flops. El dígito 1 detrás de *M* (M1) y el 1 en 1, 5 D muestran una relación de dependencia y nos indican que los datos de entrada se almacenan sólo cuando el dispositivo está en el modo de carga (M1), en el que $\overline{LOAD} = 0$. Cuando la entrada $\overline{LOAD}$ está a nivel ALTO (1), el contador avanza a través de su secuencia binaria normal, como se indica mediante M2 y el 2 en C5/2,3,4+.

Dependencia AND (G). Como muestra la Figura 9.62(b), la letra *G* denota dependencia AND, la cual indica que se efectúa una operación AND entre una entrada, designada con la letra *G* y seguida por un dígito, por un lado, y cualquier otra entrada o salida que tenga el mismo dígito como prefijo en su etiqueta. En este ejemplo en particular, G3 en la entrada *ENT* y 3CT = 15 en la salida *RCO* están relacionadas, como indica el dígito 3, y esta relación es una dependencia AND, como indica la *G*. Esto nos dice que *ENT* tiene que ser un nivel ALTO (no hay triángulo en la entrada) y que el

estado del contador debe ser quince ($CT = 15$) para que la salida *RCO* sea un nivel ALTO.

Además, los dígitos 2, 3 y 4 en la etiqueta C5/2,3,4+ indican que el contador avanza a través de sus estados cuando $\overline{LOAD} = 1$, como se indica mediante la etiqueta M2 de dependencia de modo, y cuando $ENT = 1$ y $ENP = 1$, como se indica por las etiquetas de dependencia AND G3 y G4. El signo más (+) indica que el contador se incrementa cuando existe esta condición.

Esta descripción de un símbolo lógico con notación de dependencia específico pretende ayudar en la interpretación de otros símbolos similares con los que pueda encontrarse en el futuro.

REVISIÓN DE LA SECCIÓN 9.9

1. En la notación de dependencia, ¿qué significan las letras *C*, *M* y *G*?
2. ¿Qué letra indica almacenamiento de datos?

9.10 ■ APLICACIÓN A LOS SISTEMAS DIGITALES

En esta aplicación a los sistemas digitales, vamos a continuar trabajando con el sistema de control de semáforos. Deberían revisarse las secciones 6.11 y 8.9 antes de continuar. En esta sección, el punto principal será la lógica secuencial del sistema, que constituye el último bloque del sistema que estamos desarrollando.

Al finalizar esta sección, el lector deberá ser capaz de:

- Desarrollar un diagrama de bloques para la lógica secuencial del sistema.
- Implementar el contador y la lógica de entrada, y completar el diagrama lógico del sistema.
- Comparar el diagrama lógico con la tarjeta de circuito impreso.
- Desarrollar un procedimiento de prueba y de localización de averías para la tarjeta.

Requisitos generales de la lógica secuencial

En la Figura 9.63 se presenta el diagrama global de bloques del sistema, con el bloque de lógica secuencial resaltado y la secuencia de luces. El bloque de decodificación de estados, la lógica de salida y los circuitos de temporización fueron desarrollados en los Capítulos 6 y 8. En esta sección, vamos a centrarnos en el circuito lógico que controla la secuenciación de los semáforos basándose en las entradas de los circuitos de temporización y del sensor de vehículos. La lógica secuencial produce una secuencia en código Gray de 2 bits, para los cuatro estados del sistema que se indican en la Figura 9.63.

Diagrama de bloques de la lógica secuencial

La lógica secuencial consiste en un contador en código Gray de 2 bits y la lógica de entrada asociada, como se muestra en la Figura 9.64. El contador genera una secuencia de cuatro estados. Las transiciones desde un estado al siguiente están determinadas por el intervalo de 4 s (temporizador corto) y por el intervalo de 25 s (temporizador largo) producidos por los circuitos de temporización y la entrada del sensor de vehículos. El reloj del contador es la señal de 10 kHz producida por el temporizador 555 que se encuentra en los circuitos de temporización del sistema.

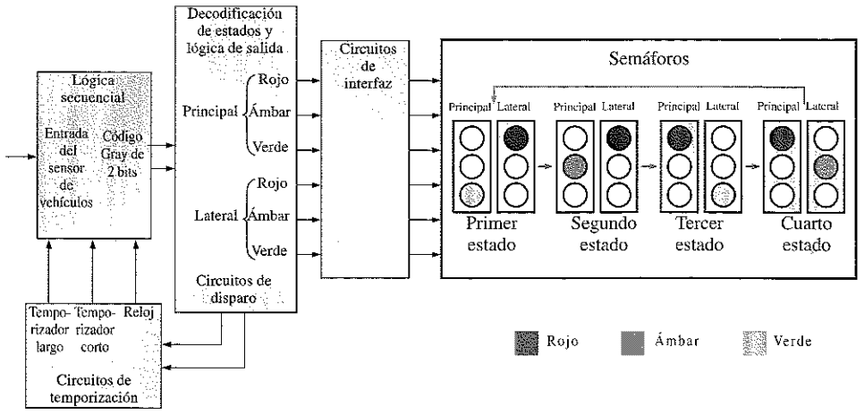


FIGURA 9.63
Diagrama de bloques del sistema de control de semáforos y secuencia de luces.

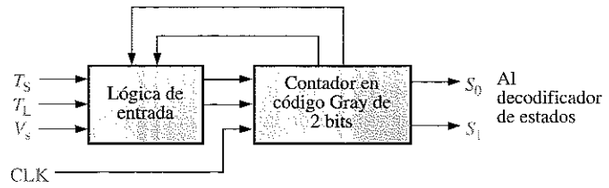


FIGURA 9.64
Diagrama de bloques de la lógica secuencial.

Diagrama de estados y funcionamiento secuencial

El diagrama de estados del sistema fue introducido en el Capítulo 6 y se muestra de nuevo en la Figura 9.65. En función de este diagrama de estados, la secuencia del código Gray se describe de la manera siguiente:

Primer estado: El código Gray para este estado es 00. La luz de la calle principal está en verde y la luz de la calle lateral está en rojo. El sistema permanece en este estado durante al menos 25 segundos, cuando el temporizador largo está *activado* o mientras que no haya vehículos en la calle secundaria ($T_L + \overline{V}_S$). El sistema pasa al estado futuro cuando el temporizador está *desactivado* y hay algún vehículo en la calle lateral ($\overline{T}_L V_S$).

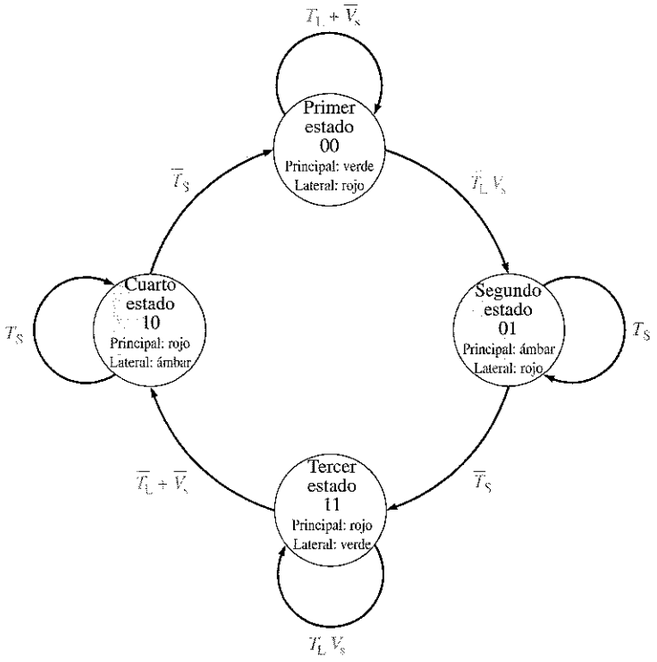


FIGURA 9.65
Diagrama de estados que muestra la secuencia de códigos Gray de 2 bits.

Segundo estado: El código Gray para este estado es 01. La luz de la calle principal está en ámbar y la luz de la calle secundaria está en rojo. El sistema permanece en este estado durante 4 segundos, mientras que el temporizador corto se encuentra *activado* (T_S), y pasa al siguiente estado cuando el mismo temporizador se *desactiva* ($\overline{T}_S$).

Tercer estado: El código Gray para este estado es 11. La luz de la calle principal está en rojo y la luz de la calle lateral está en verde. El sistema permanece en este estado mientras el temporizador largo está *activado* y hay un vehículo en la calle lateral ($T_L V_S$). El sistema pasa al siguiente estado cuando hayan transcurrido los 25 segundos o cuando no haya ningún vehículo en la calle lateral, lo primero que ocurra ($\overline{T}_L + \overline{V}_S$).

Cuarto estado: El código Gray para este estado es 10. La luz de la calle principal está en rojo y la luz de la calle lateral está en ámbar. El sistema permanece en este estado durante 4 segundos, mientras que el temporizador corto está *activado* (T_S) y vuelve al primer estado cuando el mismo temporizador se desactiva ($\overline{T}_S$).

Implementación de la lógica

El diagrama de la Figura 9.66 define la lógica secuencial correspondiente. El contador está formado por dos flip-flops D sincronizados con el reloj, cuyas entradas D provienen de la lógica de entrada. El reloj lo constituye la salida de 10 kHz del temporizador 555. La lógica de entrada tiene cinco variables de entrada: Q_1, Q_0, T_L,T_S y V_S.

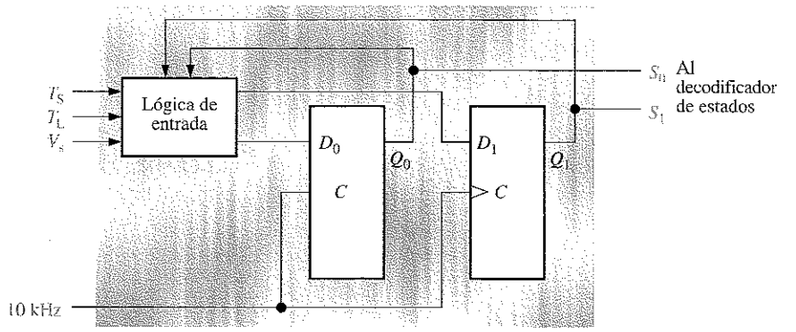


FIGURA 9.66
Lógica secuencial.

TABLA 9.13
Tabla del estado siguiente para las transiciones de la lógica secuencial.

Estado actual Q_1	Estado actual Q_0	Estado siguiente Q_1	Estado siguiente Q_0	Condiciones de entrada
0	0	0	0	$T_L + \overline{V}_s$
0	0	0	1	$\overline{T}_L V_s$
0	1	0	1	T_S
0	1	1	1	$\overline{T}_S$
1	1	1	1	$T_L V_s$
1	1	1	0	$\overline{T}_L + \overline{V}_s$
1	0	1	0	T_S
1	0	0	0	$\overline{T}_S$

TABLA 9.14
Tabla de transiciones del flip-flop D.

Transiciones de salida Q_N		Q_{N+1}	Entrada del flip-flop D
0	⟶	0	0
0	⟶	1	1
1	⟶	0	0
1	⟶	1	1

A partir del diagrama de estados, podemos desarrollar una tabla del estado siguiente, como se muestra en la Tabla 9.13. Las condiciones de entrada para T_L, T_S y V_S para cada combinación estado-actual/estado-siguiente se muestran también en la tabla. La tabla de transiciones del flip-flop D se indica en la Tabla 9.14.

■ PRÁCTICAS DE SISTEMAS

En las prácticas del Capítulo 6, estuvimos trabajando con la parte de la tarjeta controladora de los semáforos que contenía el decodificador de estados y la lógica de salida. En el Capítulo 8, fueron desarrollados los circuitos de temporización. En esta sección, vamos a completar la tarjeta, incluyendo la parte de la lógica secuencial. Al igual que antes, todas las conexiones a V_{CC} y tierra están en la parte posterior de la tarjeta y se han omitido para simplificar. Las interconexiones del circuito situadas en la parte posterior de la tarjeta, mostradas en trazo más oscuro, se conectan a través de los taladros a la cara de componentes.

■ PRÁCTICA DE SISTEMAS 1: Diseño

A partir de la Tabla 9.13 (tabla del estado siguiente) y de la Tabla 9.14 (tabla de transiciones del flip-flop D), podemos determinar las condiciones lógicas requeridas por cada flip-flop para tener un estado siguiente igual a 1. Por ejemplo, Q_0 pasa de 0 a 1 cuando el estado actual es 00 y la condición de entrada es $\overline{T}_L V_S$, como se indica en la segunda fila de la tabla que se muestra aquí. D_0 tiene que ser 1 para hacer que Q_0 pase a 1 o permanezca en 1. Esta condición para D_0 puede ser escrita como expresión lógica de la manera siguiente:

$$D_0 = \overline{Q}_1 \overline{Q}_0 \overline{T}_L V_S$$

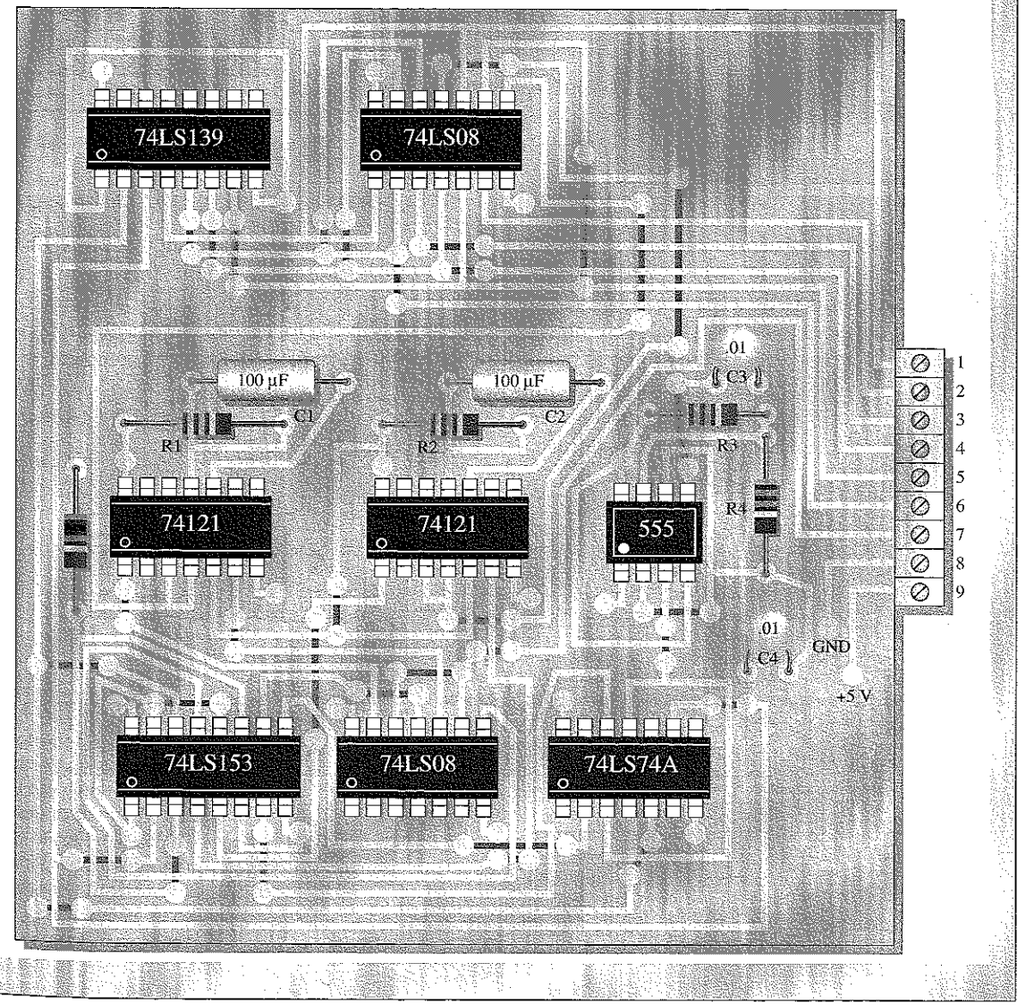
Estado actual		Estado futuro		Condiciones de entrada	Entradas de los flip-flops	
Q_1	Q_0	Q_1	Q_0		D_1	D_0
0	0	0	0	$T_L + \overline{V}_s$		
0	0	0	1	$\overline{T}_L V_s$	0	1
0	1	0	1	T_S		
0	1	1	1	$\overline{T}_S$		
1	1	1	1	$T_L V_s$		
1	1	1	0	$\overline{T}_L + \overline{V}_s$		
1	0	1	0	T_S		
1	0	0	0	$\overline{T}_S$		

- ***Actividad 1.*** Completar la parte correspondiente a las *entradas del flip-flop* en la tabla. Determinar el resto de condiciones lógicas que hacen que $D_0 = 1$ y desarrollar una expresión lógica para D_0. Conviene tener presente que estamos buscando condiciones para cada estado actual que hagan que el flip-flop pase a 1 o permanezca en 1 en el estado siguiente. Las expresiones resultantes deben facilitarse en forma de suma de productos.
- ***Actividad 2.*** Determinar todas las condiciones lógicas que hacen que $D_1 = 1$, y desarrollar una expresión lógica para D_1. Conviene tener presente que estamos buscando condiciones para cada estado actual que hagan que el flip-flop pase a 1 o permanezca en 1 en el estado siguiente. Las expresiones resultantes deben facilitarse en forma de suma de productos.
- ***Actividad 3.*** Utilizando las sumas de productos desarrolladas para D_0 y D_1, implementar la lógica secuencial a partir del diagrama de bloques de la Figura 9.64. Consultar las siguientes hojas de características:
  - Doble Flip-flop D 74LS74A
  - Cuádruple puerta AND 74LS08
  - Selector de datos (multiplexor) 74LS153

Utilizar estos dispositivos para implementar la lógica. El 74LS153 es un doble selector de datos de 4 entradas que se utiliza como generador de funciones lógicas. Este aspecto ya fue estudiado en la Sección 6.7.

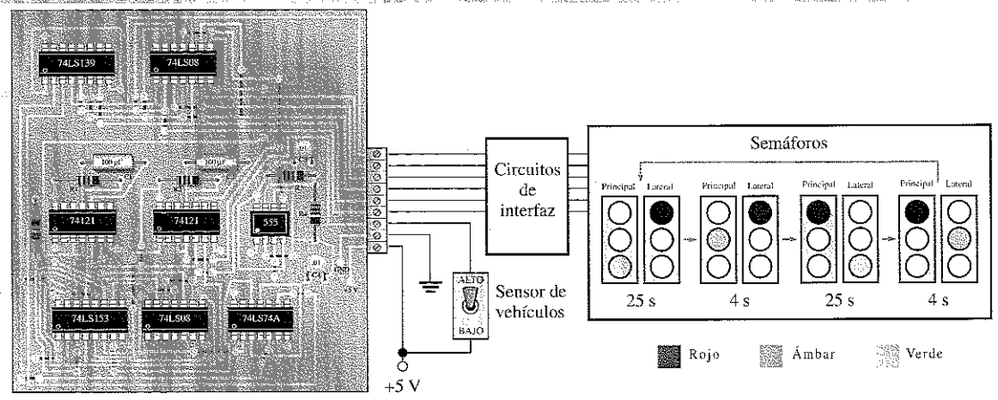
■ PRÁCTICA DE SISTEMAS 2: Verificación

- ***Actividad 1.*** Aquí se muestra la tarjeta completa del sistema, incluyendo la lógica secuencial, la decodificación de estados, la lógica de salida y los circuitos de temporización. Como muestra el diagrama de bloques del sistema, las salidas de esta tarjeta van a los circuitos de interfaz, que constituyen una parte independiente del sistema. Desarrollar el diagrama lógico del sistema completo para esta tarjeta.
- ***Actividad 2.*** Verificar que nuestro diseño de la lógica secuencial de la práctica de sistemas número 1 coincide con la circuitería de esta tarjeta.
- ***Actividad 3.*** Identificar y etiquetar las salidas y entradas del conector de la tarjeta.



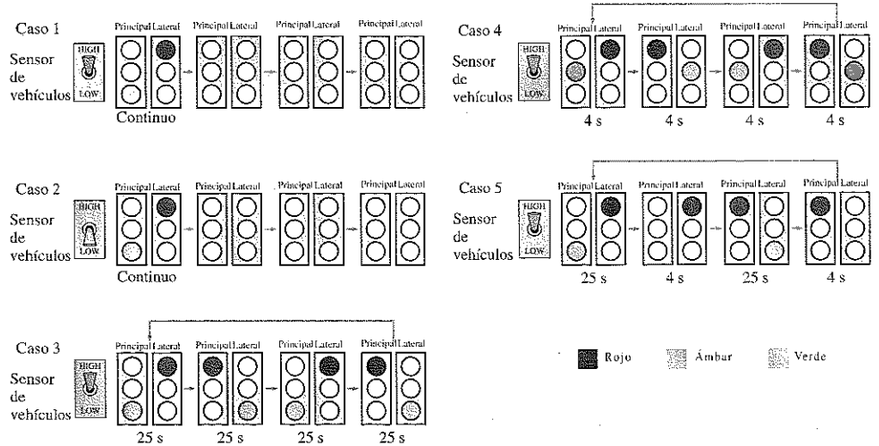
■ PRÁCTICA DE SISTEMAS 3: Localización de averías

■ ***Actividad 1.*** Desarrollar un procedimiento de pruebas para verificar la tarjeta del sistema en el laboratorio, simulando la entrada del sensor de vehículos, el circuito de interfaz y las luces de los semáforos con los componentes disponibles.



■ ***Actividad 2.*** El sistema completo se muestra en el diagrama de arriba, indicando la secuencia correcta de luces. Para cada uno de los casos, determinar si existe algún problema y, en caso de que así sea, identificar el fallo o fallos más probables en la tarjeta del circuito del sistema. Suponer que los circuitos de interfaz y los semáforos funcionan adecuadamente, de manera que cualquier problema que exista sea exclusivo de la tarjeta del sistema.

EWB La lógica de la tarjeta de circuito impreso del sistema completo se encuentra en el archivo SA09.EWB en el CD-ROM del libro. Utilícese el EWB para probar este sistema.



REVISIÓN DE LA SECCIÓN 9.10

1. Explicar el propósito de la lógica secuencial del sistema.
2. ¿Qué controla las transiciones de la lógica secuencial de un estado al siguiente?
3. Explicar por qué la frecuencia de la señal de reloj podría ser mayor o menor que 10 kHz y no afectar visualmente a la secuencia de luces.

En este punto puede estudiar el Capítulo 15, "Tecnología de circuitos integrados", parcial o totalmente. El examen de este capítulo no es un prerrequisito para abordar cualquier otro tema tratado en este texto, y su omisión total o parcial no afectará al resto de los temas.

■ RESUMEN

- Como se muestra en la Figura 9.67, los contadores síncronos y asíncronos únicamente se diferencian en la forma en que se les aplica la señal de reloj. Los contadores síncronos pueden trabajar a frecuencias de reloj mayores que los contadores asíncronos.
- En la Figura 9.68 se muestran las conexiones de los circuitos integrados contadores presentados en este capítulo.

FIGURA 9.67
Comparación de los contadores síncronos y asíncronos.

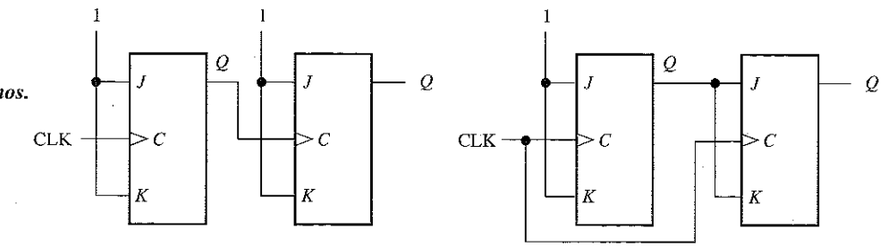
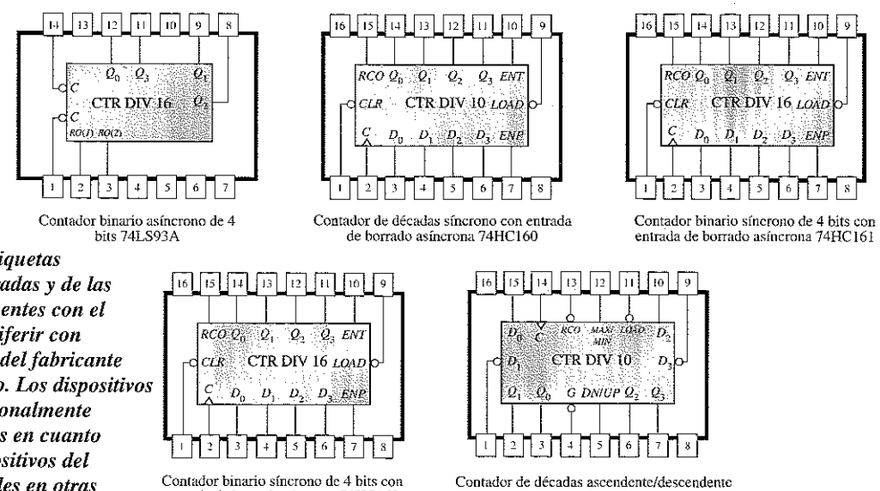


FIGURA 9.68
Obsérvese que las etiquetas (nombres de las entradas y de las salidas) son consecuentes con el texto, pero pueden diferir con respecto al catálogo del fabricante que se esté utilizando. Los dispositivos mostrados son funcionalmente iguales y compatibles en cuanto a pines con los dispositivos del mismo tipo disponibles en otras familias de circuitos integrados.



- El módulo máximo de un contador es el número máximo de posibles estados y es función del número de etapas (flip-flops). Por tanto,

$$\text{Módulo máximo} = 2^n$$

donde n es el número de etapas del contador. El módulo de un contador es el número *real* de estados de la secuencia, y puede ser igual o menor que el módulo máximo.

- El módulo global de los contadores en cascada es igual al producto de los módulos de los contadores individuales.

■ PALABRAS CLAVE

Estos términos también se encuentran en el glosario del final del libro.

Cascada Conectar dispositivos "uno tras otro", como cuando se conectan varios contadores de forma que la salida de un contador esté conectada a la entrada de habilitación del siguiente contador.

Circuito secuencial Circuito digital cuyos estados lógicos dependen de una determinada secuencia temporal.

Contador ascendente/descendente Contador que puede avanzar tanto en una dirección como en otra a lo largo de una secuencia.

Contador asíncrono Un tipo de contador en el que se utiliza como entrada de reloj de cada etapa la salida de la etapa precedente.

Contador de décadas Contador digital de diez estados.

Contador de propagación Un contador asíncrono.

Contador síncrono Un tipo de contador en el que todas las etapas utilizan el mismo impulso de reloj.

Década Que se caracteriza por diez estados o valores.

Diagrama de estados Una descripción gráfica de una secuencia de estados o valores.

Inicio de un nuevo ciclo Sufrir una transición (como ocurre en los contadores) desde el estado final o terminal hasta el estado inicial.

Máquina de estados Sistema lógico que exhibe una secuencia de estados condicionada por la lógica interna y las entradas externas. Cualquier circuito secuencial que exhibe una determinada secuencia de estados.

Secuencia Orden en que ocurren varias cosas en una determinada relación temporal.

Secuencia truncada Secuencia que no incluye todos los posibles estados de un contador.

Truncado Acortado.

Valor de fin de cuenta Estado final de la secuencia de un contador.

■ AUTOTEST

1. Los contadores asíncronos se conocen como:

(a) contadores de propagación **(b)** contadores de reloj múltiple
(c) contadores de décadas **(d)** contadores de módulo

2. Un contador asíncrono difiere de un contador síncrono en:

(a) el número de estados de su secuencia
(b) el modo de sincronización con la señal de reloj
(c) el tipo de flip-flop utilizado
(d) el valor del módulo

3. El módulo de un contador es:

(a) el número de flip-flops
(b) el número real de estados de su secuencia
(c) el número de veces que inicia un nuevo ciclo por segundo
(d) el número máximo posible de estados

4. Un contador binario de 3 bits tiene un módulo máximo de:

(a) 3 **(b)** 6 **(c)** 8 **(d)** 16

5. Un contador binario de 4 bits tiene un módulo máximo de:

(a) 16 **(b)** 32 **(c)** 8 **(d)** 4

6. El contador de módulo 12 tiene

(a) 12 flip-flops **(b)** 3 flip-flops **(c)** 4 flip-flops **(d)** temporización síncrona

7. ¿Cuál de los siguientes contadores es un ejemplo de contador con módulo truncado?

(a) Módulo 8 **(b)** Módulo 14 **(c)** Módulo 16 **(d)** Módulo 32

8. Un contador asíncrono de 4 bits formado por flip-flops tiene un retardo de propagación de la señal de reloj a Q de 12 ns. ¿Cuánto tiempo tarda el contador en iniciar un nuevo ciclo desde 1111 a 0000?

(a) 12 ns **(b)** 24 ns **(c)** 48 ns **(d)** 36 ns

9. Un contador BCD es un ejemplo de

(a) contador de módulo completo
(b) contador de décadas
(c) contador de módulo truncado
(d) (b) y (c)

10. En un contador BCD 8421, ¿cuál de los siguientes estados es un estado no válido?

(a) 1100 **(b)** 0010 **(c)** 0101 **(d)** 1000

11. Tres contadores de módulo 10 en cascada tienen un módulo global de:

(a) 30 **(b)** 100 **(c)** 1.000 **(d)** 10.000

12. Se aplica una frecuencia de reloj de 10 MHz a un contador en cascada formado por un contador de módulo 5, un contador de módulo 8, y dos contadores de módulo 10. La frecuencia de salida más baja posible es:

(a) 10 kHz **(b)** 2,5 kHz **(c)** 5 kHz **(d)** 25 kHz

13. Un contador binario ascendente/descendente de 4 bits está en el estado binario cero. El siguiente estado en el modo descendente es:

(a) 0001 **(b)** 1111 **(c)** 1000 **(d)** 1110

14. El valor de fin de cuenta de un contador binario de módulo 13 es

(a) 0000 **(b)** 1111 **(c)** 1101 **(d)** 1100

■ PROBLEMAS

SECCIÓN 9.1 Funcionamiento del contador asíncrono

1. Para el contador asíncrono de la Figura 9.69, dibujar el diagrama de tiempos completo para ocho impulsos de reloj, indicando las formas de onda de la señal de reloj, de Q_0 y de Q_1.
2. Para el contador asíncrono de la Figura 9.70, dibujar el diagrama de tiempos completo para dieciséis impulsos de reloj, indicando las formas de onda de la señal de reloj, Q_0, Q_1 y Q_2.

FIGURA 9.69

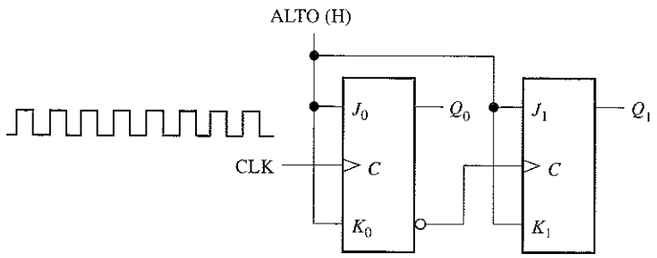
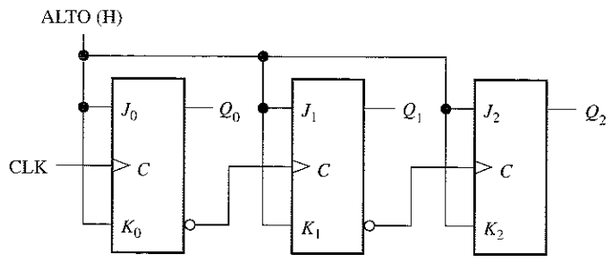


FIGURA 9.70



3. En el contador del problema 2, suponer que cada flip-flop tiene un retardo de propagación, entre el impulso de disparo de reloj y el cambio en la salida Q, de 8 ns. Determinar el retardo en el caso peor (el más largo) a partir de que se aplica un impulso de reloj hasta que el contador alcanza un determinado estado. Especificar el estado o estados para los que se produce dicho caso peor.
4. Indicar cómo se conecta un contador asíncrono de 4 bits 74LS93A, para obtener cada uno de los siguientes módulos:
   (a) 9 **(b)** 11 **(c)** 13 **(d)** 14 **(e)** 15

SECCIÓN 9.2 Funcionamiento del contador síncrono

5. Si el contador del problema 3 fuera síncrono en lugar de asíncrono, ¿cuál sería el retardo más largo?
6. Dibujar el diagrama de tiempos completo para el contador binario síncrono de la Figura 9.71. Verificar que las formas de onda de las salidas Q representan el número binario correcto después de cada impulso de reloj.
7. Analizando las entradas J y K de cada flip-flop antes de cada impulso de reloj, probar que el contador de décadas de la Figura 9.72 avanza a través de una secuencia BCD. Explicar, en cada caso, cómo estas condiciones hacen que el contador pase al siguiente estado correcto.
8. Las formas de onda de la Figura 9.73 se aplican a las entradas de reloj *(CLK)*, borrado *(CLR)* y habilitación *(CTEN)* como se indica. Dibujar las señales de salida del contador en función de estas entradas. La entrada de borrado es asíncrona.

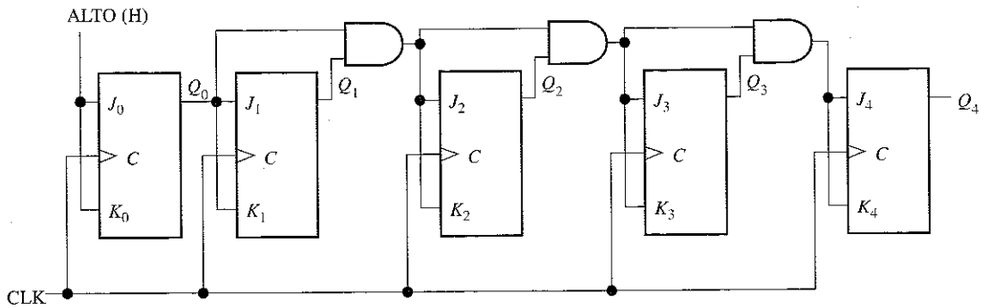


FIGURA 9.71

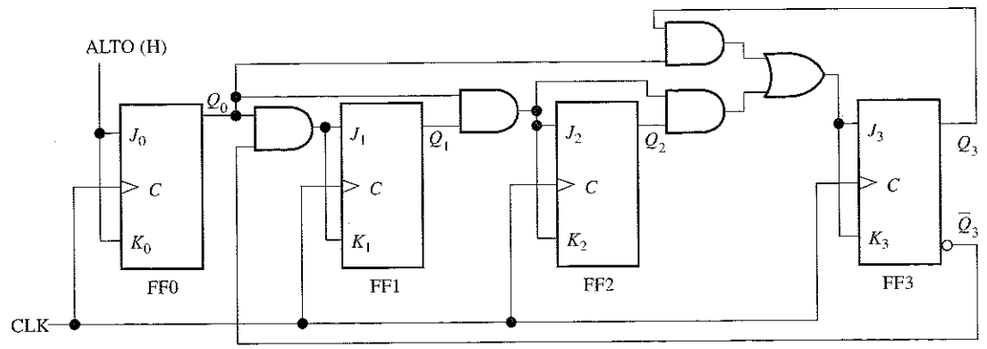
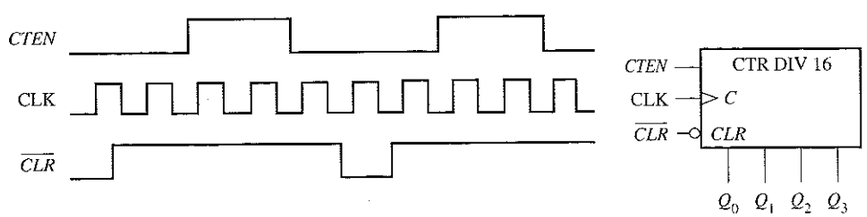


FIGURA 9.72

9. En la Figura 9.74 se muestra un contador de décadas. Se aplican las entradas de reloj *(CLK)* y de borrado *(CLR)* que se indican. Determinar las formas de onda de las salidas del contador (Q_0, Q_1, Q_2 y Q_3). La entrada de borrado es síncrona y el contador, inicialmente, está en el estado binario 1000.
10. Las formas de onda de la Figura 9.75 se aplican a un contador 74HC163. Determinar las salidas Q y *RCO*. Las entradas son $D_0 = 1$, $D_1 = 1$, $D_2 = 0$ y $D_3 = 1$.

FIGURAS 9.73



FIGURAS 9.74

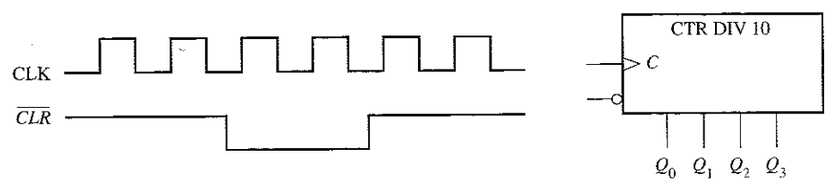
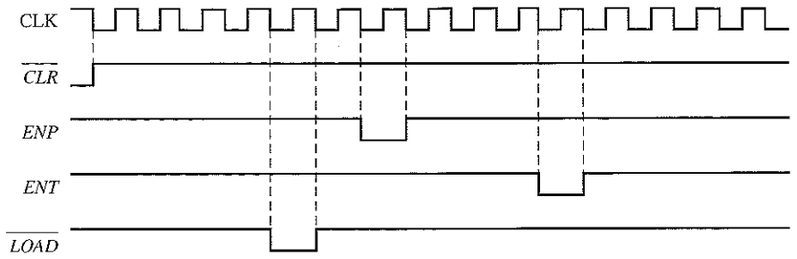


FIGURA 9.75



11. Las formas de onda de la Figura 9.75 se aplican a un contador 74HC160. Determinar las salidas Q y RCO. Las entradas son $D_0 = 1$, $D_1 = 0$, $D_2 = 0$ y $D_3 = 1$.

SECCIÓN 9.3 Contadores ascendentes/descendentes síncronos

12. Dibujar un diagrama de tiempos completo para un contador ascendente/descendente de 3 bits que sigue la siguiente secuencia. Indicar cuándo el contador está en modo ascendente y cuándo está en modo descendente. Suponer que es disparado por flanco positivo.

0, 1, 2, 3, 2, 1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1, 0

13. Dibujar la forma de onda de salida Q de un contador ascendente/descendente 74LS190 con las formas de onda de entrada mostradas en la Figura 9.76. Las entradas de datos están a cero. Comenzar la cuenta en el estado 0000.

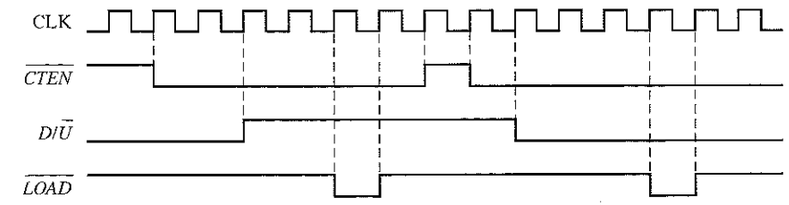


FIGURA 9.76

SECCIÓN 9.4 Diseño de los contadores síncronos

14. Determinar la secuencia del contador de la Figura 9.77.

15. Determinar la secuencia del contador de la Figura 9.78. Comenzar con el contador borrado.

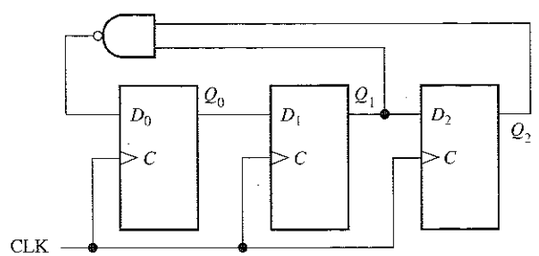


FIGURA 9.77

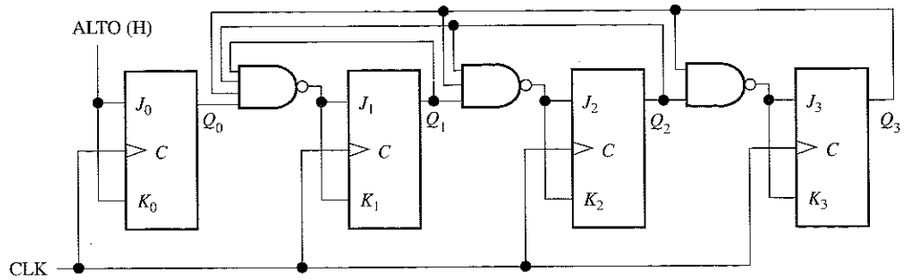


FIGURA 9.78

16. Diseñar un contador que genere la siguiente secuencia. Utilizar flip-flops J-K.

00, 10, 01, 11, 00 ...

17. Diseñar un contador que genere la siguiente secuencia binaria. Utilizar flip-flops J-K.

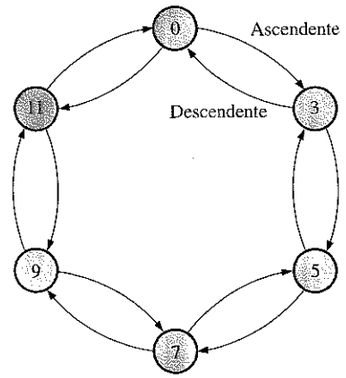
1, 4, 3, 5, 7, 6, 2, 1 ...

18. Diseñar un contador que genere la siguiente secuencia binaria. Utilizar flip-flops J-K.

0, 9, 1, 8, 2, 7, 3, 6, 4, 5, 0, ...

19. Diseñar un contador binario que genere la secuencia que indica el diagrama de estados de la Figura 9.79.

FIGURA 9.79



SECCIÓN 9.5 Contadores en cascada

20. Para cada una de las configuraciones en cascada de la Figura 9.80, determinar la frecuencia de la señal en cada punto señalado con un número encerrado en un círculo, y calcular los módulos globales.

21. Ampliar el contador de la Figura 9.41 para crear un contador divisor por 10.000 y un contador divisor por 100.000.

22. Por medio de diagramas de bloques generales, indicar cómo se obtendrán las siguientes frecuencias a partir de una señal de reloj a 10 MHz, empleando flip-flops, contadores de módulo 5 y contadores de décadas.

(a) 5 MHz **(b)** 2,5 MHz **(c)** 2 MHz **(d)** 1 MHz **(e)** 500 kHz
(f) 250 kHz **(g)** 62,5 kHz **(h)** 40 kHz **(i)** 10 kHz **(j)** 1 kHz

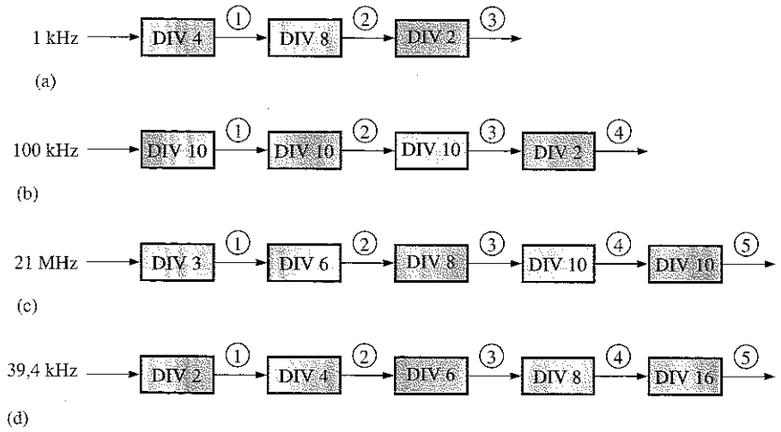


FIGURA 9.80

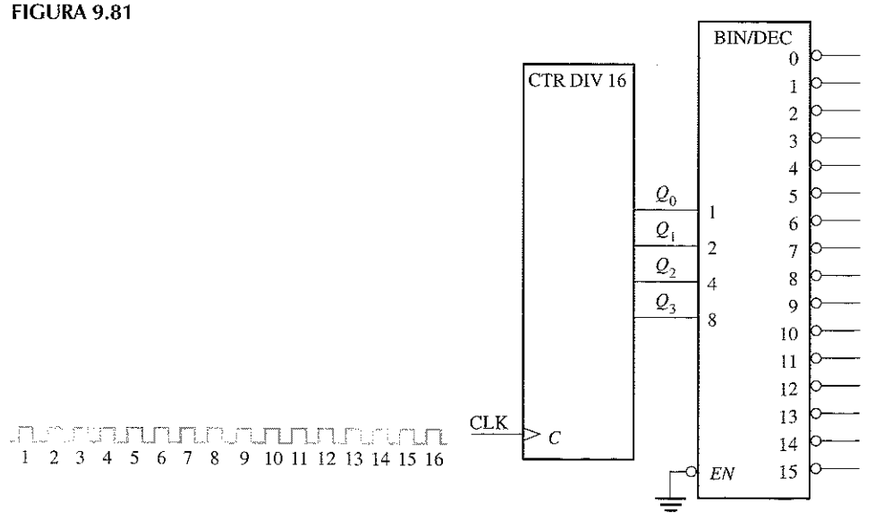
SECCIÓN 9.6 Decodificación de los contadores

23. Dado un codificador de décadas BCD con sólo disponibles las salidas Q, definir la lógica requerida para decodificar cada uno de los estados futuros e indicar cómo se conectaría al contador. Se precisa una salida a nivel ALTO para indicar cada estado decodificado. El MSB es el de la izquierda.

(a) 0001 **(b)** 0011 **(c)** 0101 **(d)** 0111 **(e)** 1000

24. Para el contador binario de 4 bits conectado al decodificador de la Figura 9.81, determinar cada forma de onda de salida del decodificador en función de los impulsos de reloj.

FIGURA 9.81



25. Si el contador de la Figura 9.81 es asíncrono, determinar dónde se producen los impulsos de ruido en las señales de salida del decodificador.

26. Modificar el circuito de la Figura 9.81 para eliminar los impulsos de ruido de decodificación.

27. Analizar la ocurrencia de glitches en la salida de la puerta de decodificación en el contador de la Figura 9.45. Si se producen glitches, sugerir una forma de eliminarlos.

28. Analizar la ocurrencia de glitches en las salidas de las puertas de decodificación en el contador de la Figura 9.46. Si éstos se producen, modificar el diseño para eliminarlos.

SECCIÓN 9.7 Aplicaciones de los contadores

29. Suponer que el reloj digital de la Figura 9.51 se inicializa a las doce horas. Determinar el estado binario de cada contador después de que se hayan producido sesenta y dos impulsos de 60 Hz de frecuencia.

30. ¿Cuál es la frecuencia de salida de cada contador en el circuito del reloj digital de la Figura 9.51?

31. Para el sistema de control del aparcamiento de coches de la Figura 9.54, en la Figura 9.82 se presenta una secuencia patrón de entrada y los impulsos del sensor para un determinado periodo de 24 horas. Si ya había 53 coches en el garaje al inicio del periodo, ¿cuál es el estado del contador pasadas las 24 horas?

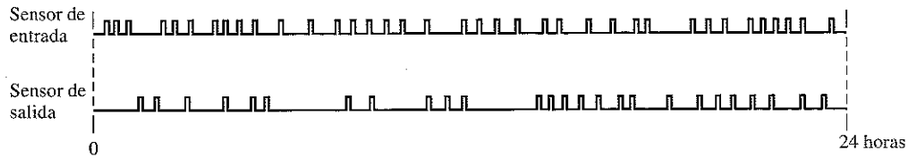


FIGURA 9.82

32. El número binario correspondiente al decimal 57 se presenta en las entradas de datos en paralelo del convertidor paralelo-serie de la Figura 9.56 (D_0 es el LSB). Inicialmente, el contador contiene todo ceros y se aplica una señal de reloj a 10 kHz. Desarrollar el diagrama de tiempos que muestre el reloj, las salidas del contador y la salida de datos serie.

SECCIÓN 9.8 Localización de averías

33. Para el contador de la Figura 9.1, dibujar el diagrama de tiempos para las formas de onda Q_0 y Q_1 si se produce alguno de los fallos siguientes (suponer que, inicialmente, Q_0 y Q_1 están a nivel BAJO):

(a) la entrada de reloj de FF0 está cortocircuitada a masa.
(b) la salida Q_0 está en circuito abierto.
(c) la entrada de reloj de FF1 está en circuito abierto
(d) la entrada J de FF0 está en circuito abierto
(e) la entrada K de FF1 está cortocircuitada a masa.

34. Resolver el problema 33 para el contador de la Figura 9.11.

35. Aislar el fallo del contador de la Figura 9.3, analizando las formas de onda de la Figura 9.83.

36. A partir del diagrama de señales de la Figura 9.84, determinar el fallo más probable en el contador de la Figura 9.14.

37. Resolver el problema 36, si la salida Q_2 se corresponde con la forma de onda de la Figura 9.85. Las salidas Q_0 y Q_1 son las de la Figura 9.84.

38. Se aplica una señal de reloj de 5 MHz al contador en cascada de la Figura 9.44 y se mide una frecuencia de 76,2939 Hz en la última salida *RCO*. ¿Es esto correcto? Si no lo es, ¿cuál es el fallo más probable?

FIGURA 9.83

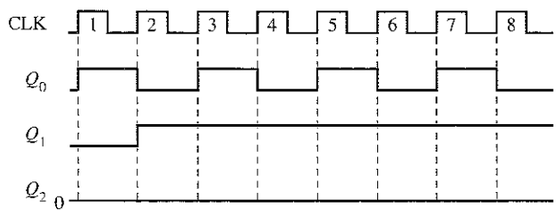


FIGURA 9.84

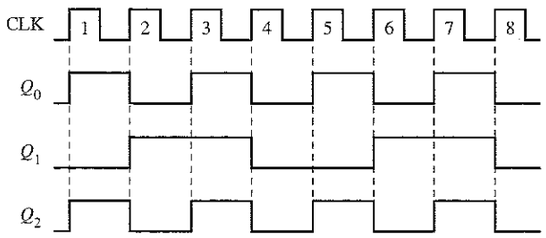
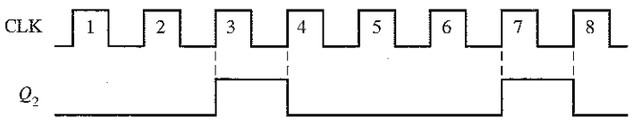


FIGURA 9.85



39. Desarrollar una tabla para probar el contador de la Figura 9.44, que muestre la frecuencia de la última salida *RCO*, para todos los posibles fallos que se producen cuando cada una de las entradas de datos (D_0, D_1, D_2 y D_3) está en circuito abierto. Utilizar una frecuencia de prueba de reloj de 10 MHz.

40. El display de 7 segmentos para las decenas de horas del sistema de reloj digital de la Figura 9.51 presenta continuamente en el display un 1. ¿Cuál es el problema?

41. ¿Cuál sería la indicación visual si la salida Q_1 del contador de decenas de minutos de la Figura 9.51 estuviera en circuito abierto? Consultar también la Figura 9.52.

42. Un determinado día (posiblemente un lunes) los dueños de las plazas del garaje que tiene el sistema de control descrito en las Figuras 9.54 y 9.55, comienzan a presentar quejas. Los dueños dicen que pueden entrar en el garaje porque la barrera está levantada y el cartel de COMPLETO está apagado, pero que una vez que han entrado no pueden encontrar una plaza vacía. Si fuera el técnico encargado de mantener este sistema, ¿cuál pensaría que era el problema? ¿Cómo localizaría la avería y repararía el sistema lo más rápidamente posible?

SECCIÓN 9.10 Aplicación a los sistemas digitales

43. Implementar la lógica de entrada del circuito secuencial del sistema de control de semáforos utilizando sólo puertas NAND.

44. Reemplazar los flip-flops D del contador en código Gray de dos bits de la Figura 9.66 por flip-flops J-K.

45. Especificar cómo se cambiaría el intervalo de la luz verde de 25 s a 60 s.

Problemas especiales de diseño

46. Diseñar un contador de módulo 1000, utilizando contadores de décadas 74HC160.

47. Modificar el diseño del contador de la Figura 9.44 para conseguir módulo 30.000.

48. Repetir el problema 47 para obtener un módulo 50.000.

49. Modificar el reloj digital de las Figuras 9.51, 9.52 y 9.53 para que pueda reinicializarse en cualquier instante.

50. Diseñar un circuito de alarma para que el reloj digital pueda detectar un intervalo de tiempo predeterminado (horas y minutos únicamente) y generar una señal que active una alarma audible.

51. Modificar el diseño del circuito de la Figura 9.55 para 1.000 y 3.000 plazas de garaje.

52. Implementar la lógica de conversión de datos paralelo-serie de la Figura 9.56 con dispositivos concretos.

53. En el problema 15, se ha determinado que el contador entra en un bucle y alterna entre dos estados. Esto sucede como resultado de un fallo de diseño. Diseñar de nuevo el contador para que cuando entre en el segundo de los estados del bucle, se inicie un nuevo ciclo en el estado de todo ceros con el siguiente impulso de reloj.

54. Modificar el diagrama de bloques del sistema de control de semáforos de la Figura 9.63, para añadir una señal de giro a la izquierda durante 15 segundos en la calle principal, inmediatamente antes de la luz verde.

Localización de fallos con el banco de trabajo electrónico

55. Abrir el archivo PR09_55.EWB y probar el contador asíncrono de 4 bits para determinar si existe un fallo. En caso afirmativo, identificarlo si es posible.

56. Abrir el archivo PR09_56.EWB y probar el contador síncrono de 3 bits para determinar si existe un fallo. En caso afirmativo, identificarlo si es posible.

57. Abrir el archivo PR09_57.EWB y probar el contador BCD para determinar si existe un fallo. En caso afirmativo, identificarlo si es posible.

58. Abrir el archivo PR09_58.EWB y probar el contador binario de 4 bits 74163 para determinar si existe un fallo. En caso afirmativo, identificarlo si es posible.

59. Abrir el archivo PR09_59.EWB y probar el contador de décadas ascendente/descendente 74190 para determinar si existe un fallo. En caso afirmativo, identificarlo si es posible.

■ RESPUESTAS A LAS REVISIONES DE CADA SECCIÓN

SECCIÓN 9.1

1. Asíncrono significa que cada flip-flop posterior al primero se activa por medio de la salida del flip-flop precedente.
2. Un contador de módulo 14 tiene catorce estados, requiriéndose cuatro flip-flops.

SECCIÓN 9.2

1. Todos los flip-flops de un contador síncrono se sincronizan simultáneamente con la señal de reloj.
2. El contador se puede inicializar en cualquier estado.
3. El contador se activa cuando *ENP* y *ENT* están a nivel ALTO; *RCO* pasa a nivel ALTO cuando se alcanza el estado final de la secuencia.

SECCIÓN 9.3

1. El contador pasa al estado 1001
2. ASCENDENTE: 1111, DESCENDENTE: 0000; el siguiente estado es 1111.

SECCIÓN 9.4

1. $J = 1$, $K =$ X (indiferente)
2. $J = X$ (indiferente), $K = 0$
3. **(a)** El estado siguiente es 1011
   (b) Q_3 (MSB): modo no cambio o SET; Q_2: modo no cambio o RESET; Q_1: modo no cambio o SET; Q_0 (LSB): modo SET o de basculación.

SECCIÓN 9.5

1. Tres contadores de décadas producen ÷1000, cuatro contadores de décadas producen ÷10.000.
2. **(a)** ÷20: flip-flop y divisor por 10
   (b) ÷32: flip-flop y divisor por 16
   (c) ÷160: divisor por 16 y divisor por 10
   (d) ÷320: divisor por 16, divisor por 10 y flip-flop.

SECCIÓN 9.6

1. **(a)** No hay ningún estado transitorio, porque hay un único cambio de bit.
   (b) 0000, 0001, 0010, 0101, 0110, 0111
   (c) No hay ningún estado transitorio, porque hay un único cambio de bit.
   (d) 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110

SECCIÓN 9.7

1. La puerta G_1 pone a cero el flip-flop en el primer impulso de reloj después de alcanzar el valor 12. La puerta G_2 decodifica el valor 12 para inicializar el contador a 0001.
2. El contador de décadas para las horas avanza a través de cada estado desde cero hasta nueve y, al pasar de nueve a cero para iniciar un nuevo ciclo, el flip-flop bascula al estado SET. Esto da lugar a que se presente un 10 en el display. Cuando el contador de décadas para las horas está en el estado 12, las puertas de decodificación NAND hacen que el contador inicie un nuevo ciclo en el estado 1 con el siguiente impulso de reloj. El flip-flop pasa a estado RESET. Esto hace que aparezca un 1 (01) en el display.

SECCIÓN 9.8

1. No hay impulsos en las salidas *TC*: *CTEN* del primer contador está cortocircuitada a masa o a un nivel BAJO; la entrada de reloj del primer contador está en circuito abierto; la línea de reloj está cortocircuitada a masa o a un nivel BAJO; la salida *TC* del primer contador está cortocircuitada a masa o a un nivel BAJO.
2. Con la salida del inversor en circuito abierto, el contador no puede comenzar un nuevo ciclo en el valor de carga predeterminado, sino que actúa como un contador de módulo completo.

SECCIÓN 9.9

1. *C*: control, usualmente reloj; *M*: modo; *G*: AND
2. *D* indica almacenamiento de datos

SECCIÓN 9.10

1. La lógica secuencial genera la secuencia de estados adecuada para controlar las luces.
2. El temporizador largo, el temporizador corto y/o el sensor de vehículos controlan las transiciones lógicas secuenciales.
3. El intervalo de 4 s es el más pequeño y es mucho mayor que el periodo de una señal de reloj de 10 kHz.

■ RESPUESTAS A LOS PROBLEMAS RELACIONADOS DE LOS EJEMPLOS

9.1. Véase la Figura 9.86.

9.2. Conectar Q_0 a la puerta NAND como tercera entrada (Q_2 y Q_3 son dos de las entradas). Conectar la línea $\overline{CLR}$ a la entrada $\overline{CLR}$ de FF0, así como de FF2 y FF3.

9.3. Véase la Figura 9.87.

9.4. Véase la Figura 9.88.

FIGURA 9.86

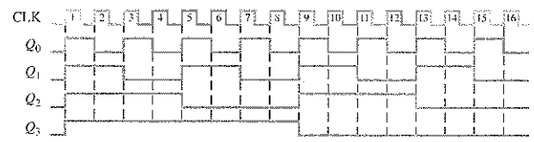


FIGURA 9.87

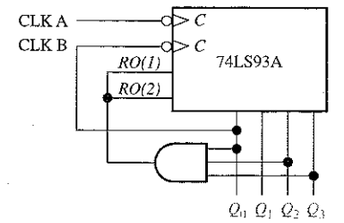


FIGURA 9.88

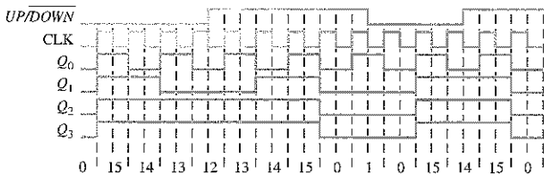


TABLA 9.15

Estado no válido actual			Entradas *J-K*						Estado siguiente			
Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0	Q_2	Q_1	Q_0	
0	0	0	0	0	1	1	1	1	0	1	1	
0	1	1	1	1	1	1	1	1	1	0	0	
1	0	0	0	0	1	1	1	0	1	1	1	estado válido
1	1	0	1	1	1	1	1	0	0	0	1	estado válido

9.5. Véase la Tabla 9.5.

9.6. La aplicación del álgebra de Boole a la lógica de la figura 9.37 demuestra que la salida de cada puerta OR está de acuerdo con la expresión del paso 5.

9.7. Se requieren contadores de cinco décadas. $10^5 = 100.000$

9.8. $f_{Q0} = 1$ MHz/[(10)(2)] = 50 kHz

9.9. Véase la Figura 9.89.

9.10. Debería cargarse $8AC0_{16}$.

$16^4 - 8AC0_{16} = 65.536 - 32.520 = 30.016$

$f_{TC4} = 10\ MHz/30,016 = 333,2\ Hz$

9.11. Véase la Figura 9.90.

FIGURA 9.89

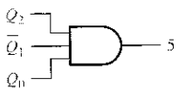
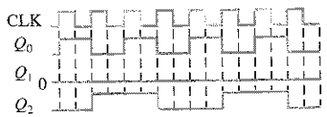


FIGURA 9.90



■ RESPUESTAS AL AUTOTEST

1. (a) **2.** (b) **3.** (b) **4.** (c) **5.** (a) **6.** (c) **7.** (b) **8.** (c)
9. (d) **10.** (a) **11.** (c) **12.** (b) **13.** (b) **14.** (d)

10

REGISTROS DE DESPLAZAMIENTO

10.1 Funciones básicas de los registros de desplazamiento
10.2 Registros de desplazamiento con entrada y salida serie
10.3 Registros de desplazamiento con entrada serie/salida paralelo
10.4 Registros de desplazamiento con entrada paralelo/salida serie
10.5 Registros de desplazamiento con entrada y salida paralelo
10.6 Registros de desplazamiento bidireccionales
10.7 Contadores basados en registro de desplazamiento
10.8 Aplicaciones de los registros de desplazamiento
10.9 Localización de averías
10.10 Símbolos lógicos con notación de dependencia
10.11 Aplicación a los sistemas digitales

WWW Simulación por computadora de circuitos digitales:
El tutorial de Electronics Workbench (EWB) se encuentra en la dirección http://www.prenhall.com/floyd.

■ OBJETIVOS DEL CAPÍTULO

- ❑ Identificar las formas básicas de movimiento de datos en los registros de desplazamiento.
- ❑ Explicar cómo funcionan los registros de desplazamiento con: entrada y salida serie, entrada serie y salida paralelo, entrada paralelo y salida serie, entrada y salida paralelo.
- ❑ Describir cómo funciona un registro de desplazamiento bidireccional.
- ❑ Determinar la secuencia de un contador Johnson.
- ❑ Configurar un contador en anillo para generar una secuencia específica.
- ❑ Realizar un contador en anillo a partir de un registro de desplazamiento.
- ❑ Utilizar un registro de desplazamiento como dispositivo de retardo de tiempo.
- ❑ Utilizar un registro de desplazamiento para implementar un convertidor de datos serie-paralelo.
- ❑ Implementar un codificador de teclado básico controlado por un registro de desplazamiento.
- ❑ Localizar averías en los sistemas probando el sistema con una secuencia patrón de pruebas conocida.
- ❑ Interpretar los símbolos estándar ANSI/IEEE 91-1984 de los registros de desplazamiento con notación de dependencia.
- ❑ Utilizar los registros de desplazamiento en un sistema digital.

■ INTRODUCCIÓN

Los registros de desplazamiento son circuitos lógicos secuenciales, que están íntimamente relacionados con los contadores digitales. Los registros se utilizan principalmente para almacenar datos digitales y, normalmente, no poseen una secuencia característica interna de estados como los contadores. Sin embargo, existen excepciones, que abordaremos en la Sección 10.7.

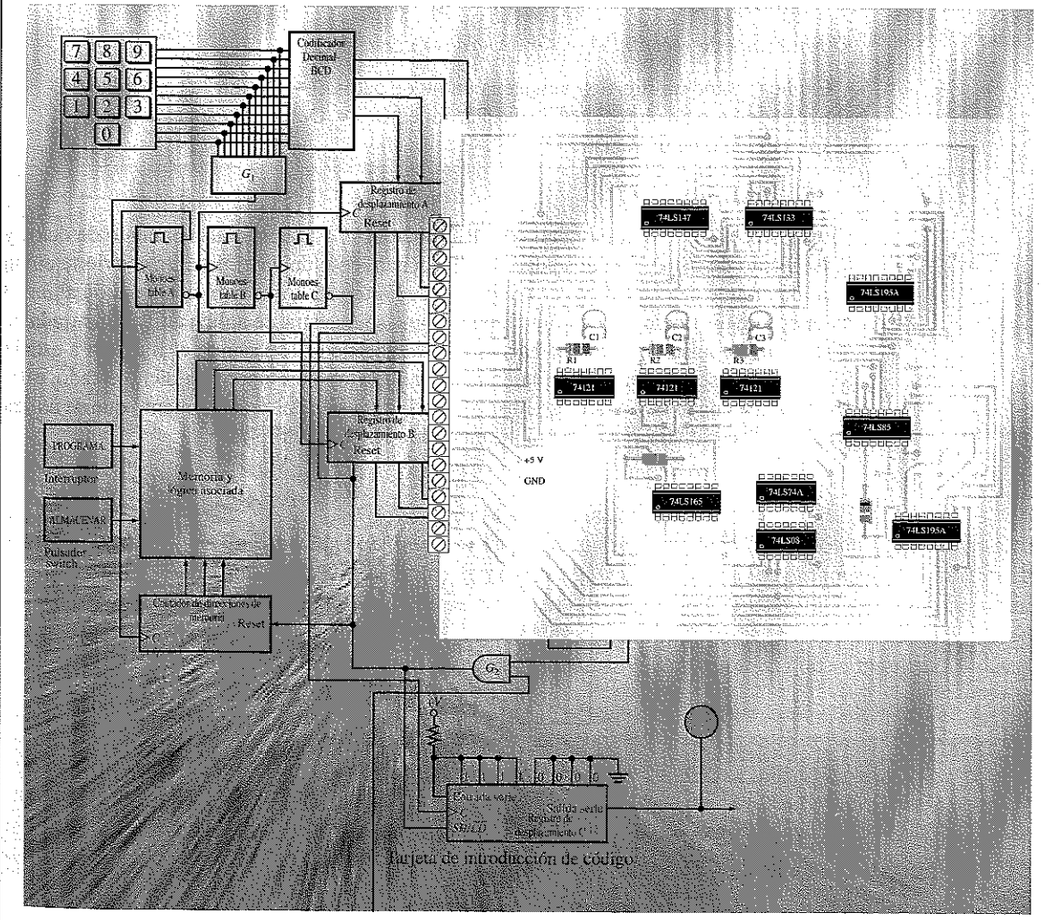
En este capítulo, se estudian los tipos básicos de registros de desplazamiento y se presentan diversas aplicaciones. También se introduce un importante método para la localización de averías.

DISPOSITIVOS ESPECÍFICOS

74HC164 74HC165 74HC174A
74HC194 74HC195

■ APLICACIÓN A UN SISTEMA DIGITAL

Esta aplicación sobre sistemas digitales ilustra los conceptos que se aprenderán en este capítulo. El sistema digital de la Sección 10.11 introduce un sistema de seguridad para acceso, que controla las alarmas de un edificio. Este sistema utiliza dos tipos de registros de desplazamiento, así como otros dispositivos que se han visto en los capítulos anteriores. El sistema también incluye una memoria, dispositivo que se tratará en el Capítulo 12.



10.1 ■ FUNCIONES BÁSICAS DE LOS REGISTROS DE DESPLAZAMIENTO

Los registros de desplazamiento están formados por un conjunto de flip-flops, y son muy importantes en las aplicaciones que precisan almacenar y transferir datos dentro de un sistema digital. La diferencia básica entre un registro y un contador es que un registro no tiene una secuencia de estados específica, excepto en ciertas aplicaciones muy especializadas. En general, un registro se utiliza únicamente para almacenar y desplazar datos (1s y 0s), que introduce en él una fuente externa y, normalmente, no posee ninguna secuencia característica interna de estados.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar cómo un flip-flop almacena un bit de datos. ■ Definir la capacidad de almacenamiento de un registro de desplazamiento. ■ Definir la capacidad de desplazamiento de un registro.

Un **registro** es un circuito digital con dos funciones básicas: almacenamiento de datos y movimiento de datos. La capacidad de almacenamiento de un registro le convierte en un tipo importante de dispositivo de *memoria*. La Figura 10.1 ilustra el concepto de almacenamiento de un 1 o un 0 en un flip-flop D. Como se muestra, se aplica un 1 a la entrada de datos y un impulso de reloj que hace que se almacene el 1, pasando el flip-flop a estado SET. Cuando se elimina el 1 de la entrada, el flip-flop permanece en dicho estado, quedando almacenado el 1. Como se ilustra en la Figura 10.1, el procedimiento que se utiliza para almacenar un 0 es similar y pone en estado RESET al flip-flop.

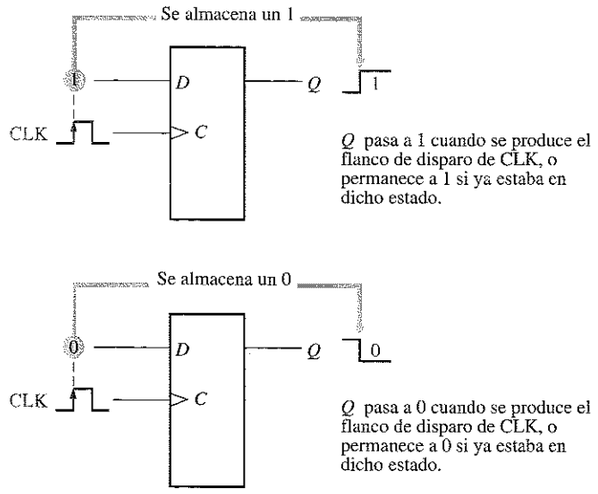


FIGURA 10.1
El flip-flop como elemento de almacenamiento.

La *capacidad de almacenamiento* de un registro es el número total de bits (1s y 0s) de un dato digital que puede contener. Cada **etapa** (flip-flop) de un registro de **desplazamiento** representa un bit de su capacidad de almacenamiento; por tanto, el número de etapas de un registro determina su capacidad de almacenamiento.

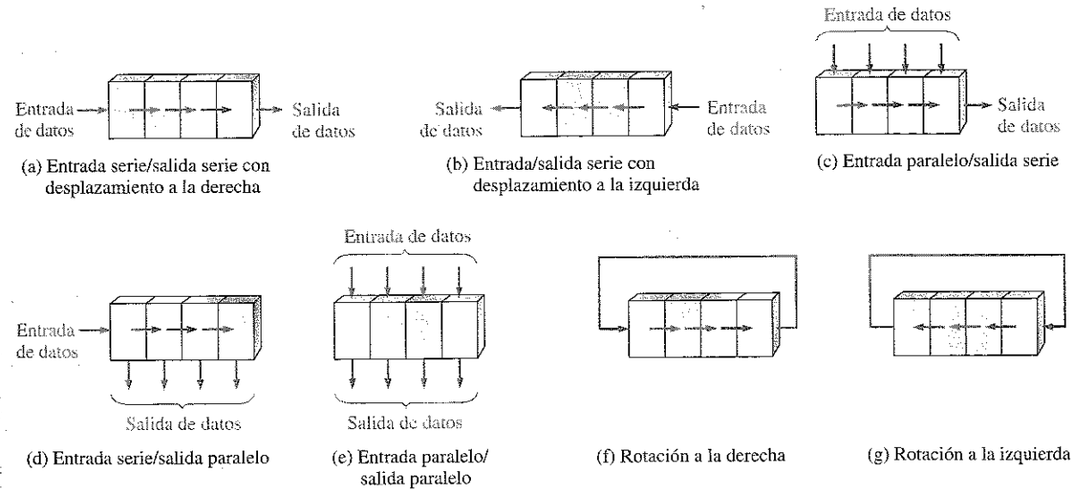


FIGURA 10.2
Movimientos básicos de datos en los registros de desplazamiento (en este ejemplo se emplean cuatro bits).

La capacidad de desplazamiento de un registro permite el movimiento de los datos de una etapa a otra dentro del registro, o la entrada o salida del mismo, en función de los impulsos de reloj que se apliquen. La Figura 10.2 ilustra la forma en que se pueden mover los datos en los registros de desplazamiento. El bloque representa un registro cualquiera de 4 bits y las flechas indican la dirección en que se mueven los datos.

REVISIÓN DE LA SECCIÓN 10.1

1. Generalmente, ¿cuál es la diferencia entre un contador y un registro de desplazamiento?
2. ¿Cuáles son las dos principales funciones que realiza un registro de desplazamiento?

10.2 ■ REGISTROS DE DESPLAZAMIENTO CON ENTRADA / SALIDA SERIE

Los registros de desplazamiento con entrada y salida serie aceptan datos en serie, es decir, un bit cada vez por una única línea. La información almacenada es entregada a la salida también en forma serie.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar cómo se introducen en serie los bits de datos en un registro de desplazamiento. ■ Describir cómo se desplazan los bits de datos a través del registro.

- Explicar cómo los bits de datos salen en serie del registro de desplazamiento.
- Desarrollar y analizar los diagramas de tiempos para los registros con entrada y salida serie.

NOTAS INFORMÁTICAS
Frecuentemente, es necesario *borrar* un registro interno de una computadora. Por ejemplo, un registro puede borrarse antes de realizar una operación aritmética o de otro tipo. Un método para borrar los registros de una computadora consiste en utilizar software para extraer los contenidos del registro. Por supuesto, el resultado siempre será cero. Por ejemplo, una instrucción que realiza esta operación es SUB AL,AL. Con esta instrucción, el registro denominado AL se borrará.

En primer lugar, vamos a ver la introducción en serie de datos en un registro de desplazamiento típico. La Figura 10.3 muestra un dispositivo de 4 bits implementado con flip-flops D. Con cuatro etapas, este registro puede almacenar hasta cuatro bits de datos.

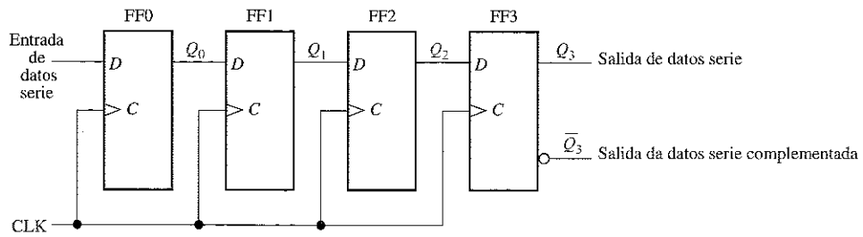


FIGURA 10.3
Registro de desplazamiento con entrada / salida serie.

La Figura 10.4 ilustra la introducción en el registro de cuatro bits, 1010, comenzando por el bit más a la derecha. Inicialmente, el registro se borra (CLEAR). Se aplica un 0 en la línea de entrada de datos, lo que hace $D = 0$ en el flip-flop FF0. Cuando se aplica el primer impulso de reloj, FF0 pasa al estado RESET, almacenando el 0.

A continuación se aplica a la entrada de datos el segundo bit que, en este caso, es 1, lo que hace que $D = 1$ en FF0 y $D = 0$ en FF1 debido a que la entrada D de FF1 está conectada a la salida Q_0. Cuando se produce el segundo impulso de reloj, el 1 de la entrada de datos de FF0 se desplaza, pasando este flip-flop al estado SET, y el 0 que había en FF0 se desplaza a FF1.

El tercer bit, un 0, se introduce por la línea de entrada de datos y se aplica un impulso de reloj. El 0 entra en FF0, el 1 almacenado en éste se desplaza a FF1 y el 0 almacenado en FF1 se desplaza a FF2.

El último bit, que es un 1, se aplica a la entrada de datos y se aplica el siguiente impulso de reloj. Ahora el 1 entra en FF0, el 0 almacenado en éste se desplaza a FF1, el 1 almacenado en FF1 se desplaza a FF2, y el 0 almacenado en FF2 se desplaza a FF3. Esto completa la introducción en serie de los cuatro bits en el registro de desplazamiento, donde pueden quedar almacenados el tiempo que se desee, siempre que los flip-flops estén alimentados con la tensión continua necesaria.

Si se desea extraer los datos del registro, los bits deben desplazarse en serie hasta la salida Q_3, como se ilustra en la Figura 10.5. Después del cuarto impulso de reloj CLK4, el bit más a la derecha, 0, está en la salida Q_3. Si se aplica un quinto impulso de reloj, CLK5, el segundo bit aparecerá en la salida Q_3. El impulso de reloj CLK6 desplaza el tercer bit a la salida y el séptimo impulso de reloj (CLK7) desplaza el cuarto bit. Obsérvese que, mientras que los cuatro bits iniciales se desplazan a la salida, se pueden introducir otros bits de datos. En la figura se muestra cómo se ha desplazado una serie de ceros.

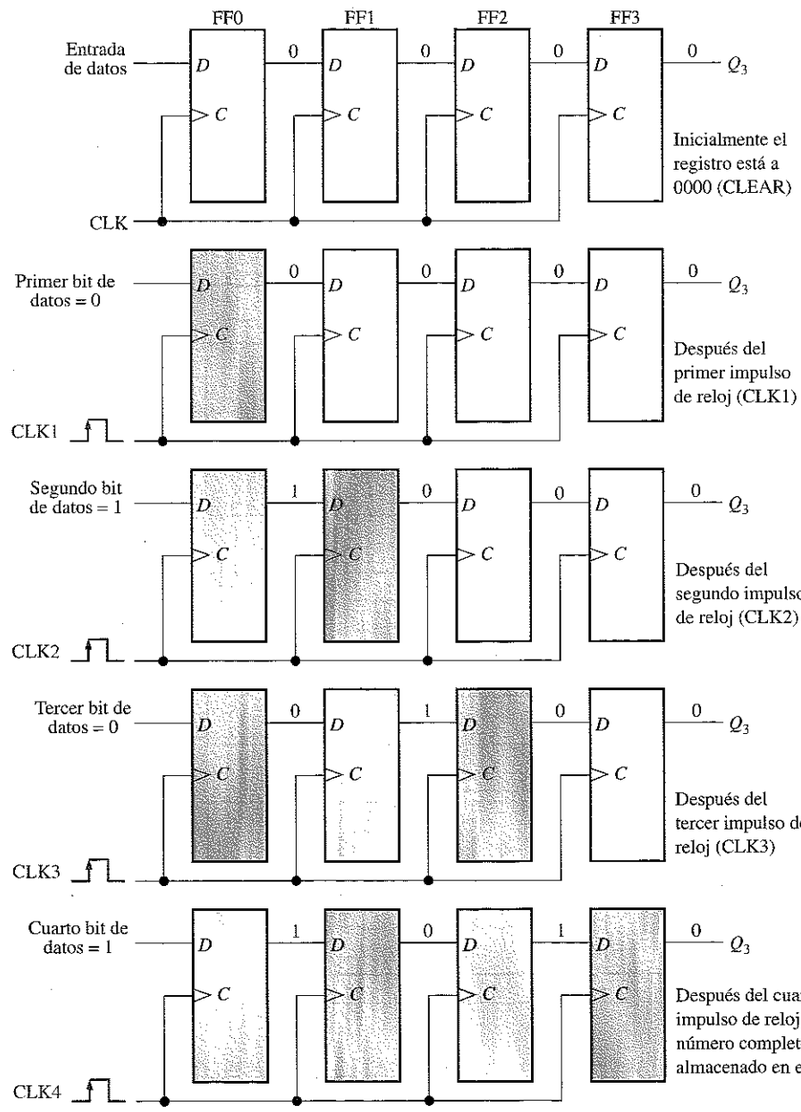


FIGURA 10.4
Introducción de cuatro bits en serie (1010) en el registro.

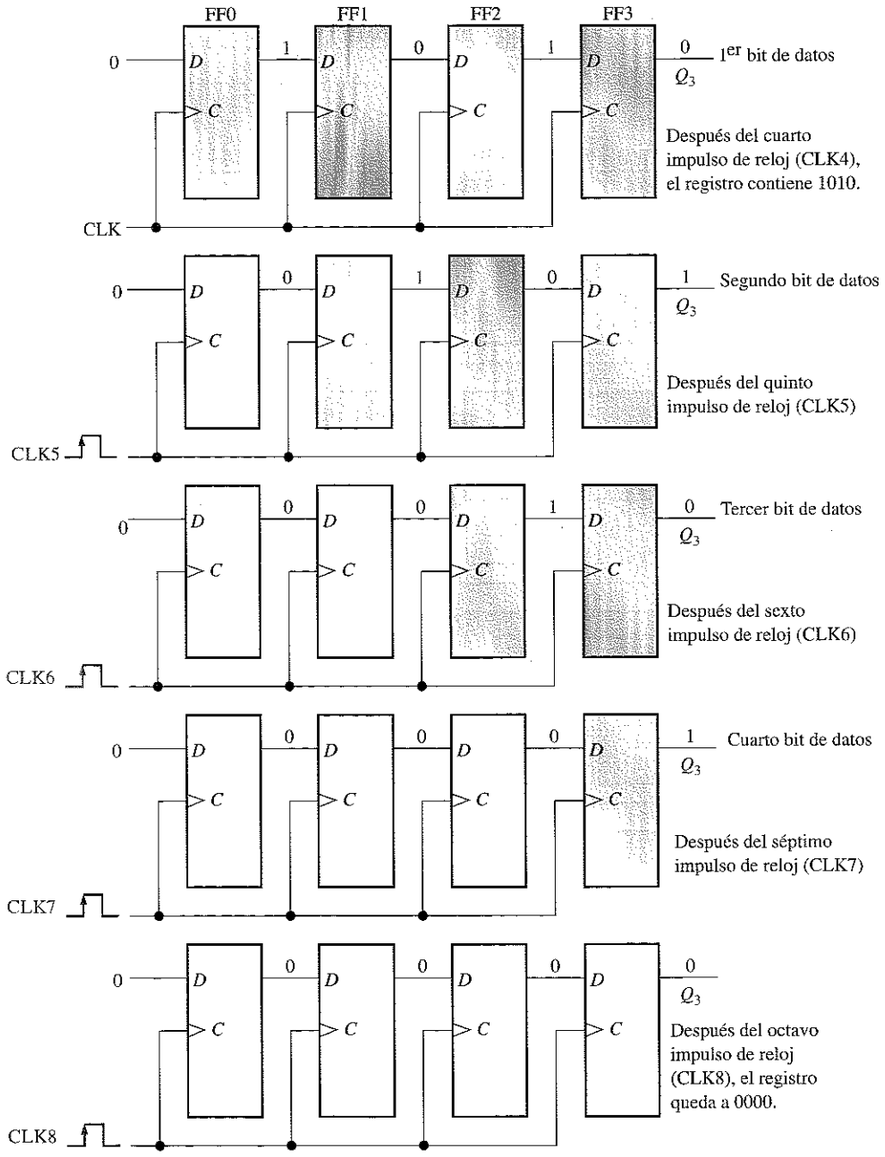


FIGURA 10.5
Los cuatro bits (1010) se han desplazado en serie a la salida del registro y se han reemplazado por ceros.

EJEMPLO 10.1 Obtener los estados del registro de 5 bits de la Figura 10.6(a) para las señales de entrada de datos y la señal de reloj indicadas. Suponer que, inicialmente, el registro se ha borrado (su contenido es todo ceros).

Solución. Se introduce el primer bit de datos (1) en el registro con el primer impulso de reloj y luego se desplaza de izquierda a derecha. Del mismo modo se introducen y desplazan los restantes bits. Después de cinco impulsos de reloj el registro contiene $Q_4Q_3Q_2Q_1Q_0 = 11010$. Véase la Figura 10.6(b).

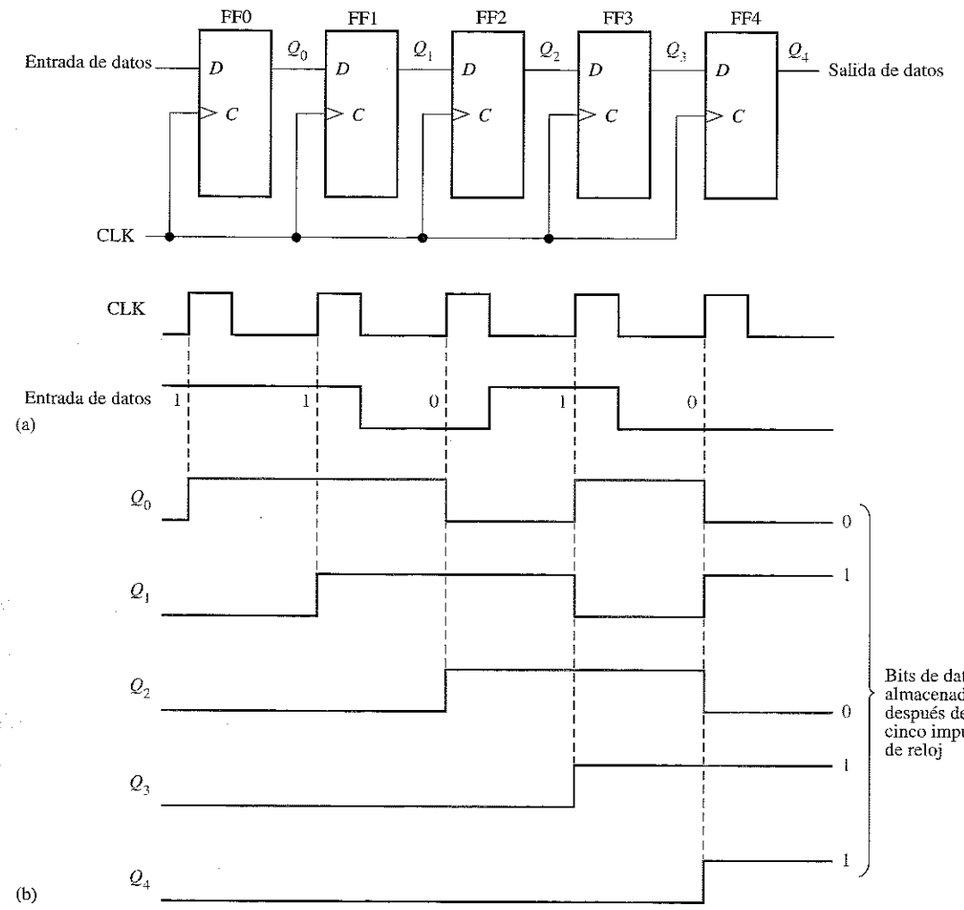
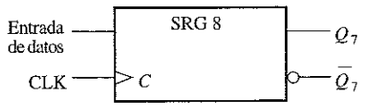


FIGURA 10.6

Problema relacionado. Obtener los estados del registro si se invierte la entrada de datos. Inicialmente el registro se borra.

En la Figura 10.7 se muestra el símbolo lógico tradicional de un registro de desplazamiento de 8 bits con entrada / salida serie. La designación "SRG8" indica que es un registro de desplazamiento (SRG, Shift Register) con una capacidad de 8 bits.

FIGURA 10.7
Símbolo lógico de un registro de desplazamiento de 8 bits con entrada / salida serie.



REVISIÓN DE LA SECCIÓN 10.2

1. Desarrollar el diagrama lógico del registro de desplazamiento de la Figura 10.3, utilizando flip-flops J-K que reemplacen a los flip-flops D.
2. ¿Cuántos impulsos de reloj se requieren para introducir un byte de datos en serie en un registro de desplazamiento de 8 bits?

10.3 ■ REGISTROS DE DESPLAZAMIENTO CON ENTRADA SERIE/SALIDA PARALELO

En este tipo de registro los bits de datos se introducen en serie (empezando por el bit situado más a la derecha), del mismo modo que se ha visto en la Sección 10.2. La diferencia está en la forma en que dichos bits se extraen del registro; en un registro con salida paralelo, se dispone de la salida de cada etapa. Una vez que los datos se han almacenado, cada bit se presenta en su respectiva línea de salida, estando disponibles todos los bits simultáneamente, en lugar de bit a bit como en el caso de la salida serie.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar cómo se extraen los bits de datos en un registro de desplazamiento con salida paralelo. ■ Comparar la salida serie y la salida paralelo. ■ Utilizar el registro de desplazamiento de 8 bits 74HC164. ■ Desarrollar y analizar los diagramas de tiempos de los registros con entrada serie-salida paralelo.

La Figura 10.8 muestra un registro de desplazamiento con entrada serie/salida paralelo, y su símbolo lógico.

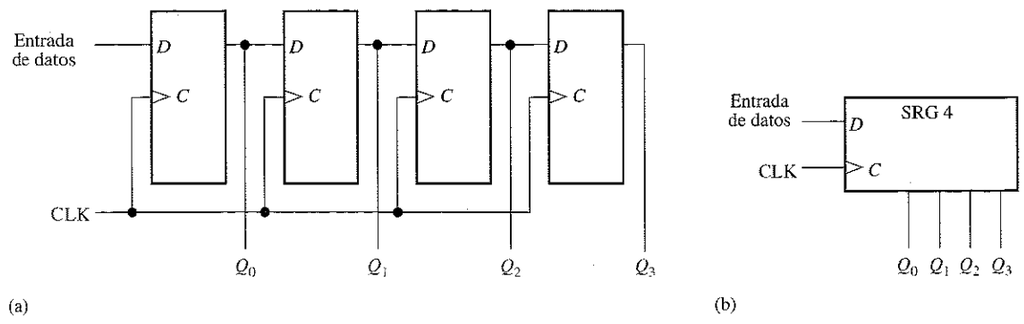


FIGURA 10.8
Registro de desplazamiento con entrada serie/salida paralelo.

EJEMPLO 10.2 Determinar los estados del registro de 4 bits (SRG 4) para las señales de entrada de datos y de reloj de la Figura 10.9(a). Inicialmente, el contenido del registro es todo unos.

Solución. Después de cuatro impulsos de reloj, el registro está en el estado 0110. Véase la Figura 10.9(b).

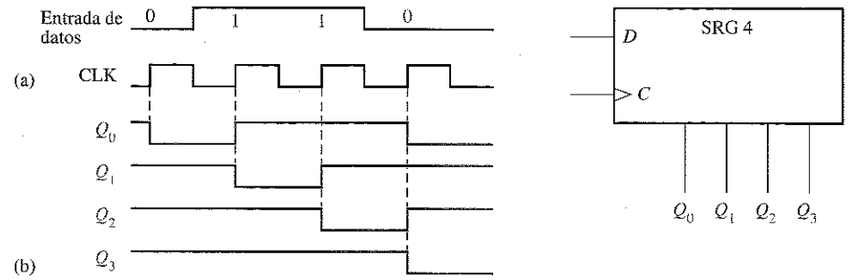


FIGURA 10.9

Problema relacionado. Si la entrada de datos después del cuarto impulso de reloj se mantiene a 0, ¿cuál será el estado del registro después de tres impulsos de reloj más?

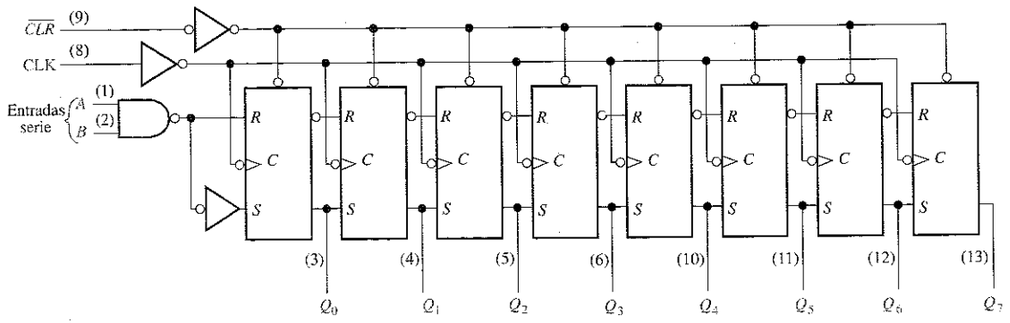
Un registro de desplazamiento de 8 bits con entrada serie/salida paralelo

El 74HC164 es un ejemplo de registro de desplazamiento en formato de circuito integrado, que funciona con entrada serie/salida paralelo. En la Figura 10.10(a) se muestra su esquema lógico y en la parte (b) el símbolo lógico típico. Obsérvese que este dispositivo dispone de dos entradas serie, *A* y *B*, a través de una puerta, y una entrada de borrado activa a nivel BAJO ($\overline{CLR}$). Las salidas paralelo son Q_0 hasta Q_7.

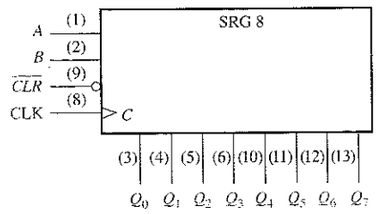
En la Figura 10.11 se muestra un sencillo diagrama de tiempos para el 74HC164. Obsérvese que los datos de entrada serie de la entrada *A* se desplazan al interior y a través del registro después de que la entrada *B* pasa a nivel ALTO.

REVISIÓN DE LA SECCIÓN 10.3

1. La secuencia de bits 1101 se introduce en serie (primero el bit más a la derecha) en un registro de desplazamiento de 4 bits con salida paralelo, que inicialmente se ha borrado. ¿Cuáles son las salidas *Q* después de dos impulsos de reloj?
2. ¿Puede utilizarse un registro con entrada serie/salida paralelo como registro con entrada y salida serie?



(a) Diagrama lógico



(b) Símbolo lógico

FIGURA 10.10
El registro de desplazamiento de 8 bits con entrada serie/salida paralelo 74HC164.

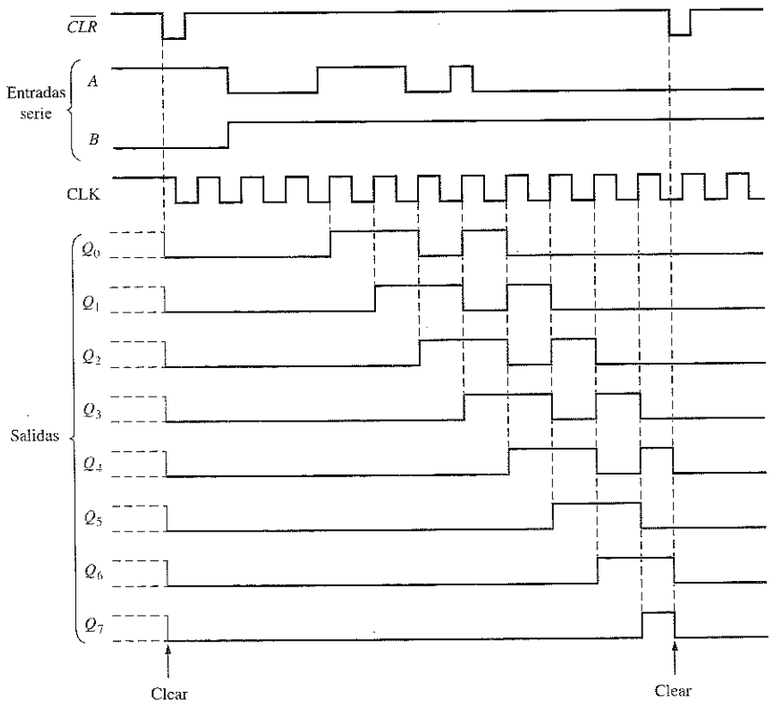


FIGURA 10.11
Diagrama de tiempos de un registro de desplazamiento 74HC164.

10.4 ■ REGISTROS DE DESPLAZAMIENTO CON ENTRADA PARALELO / SALIDA SERIE

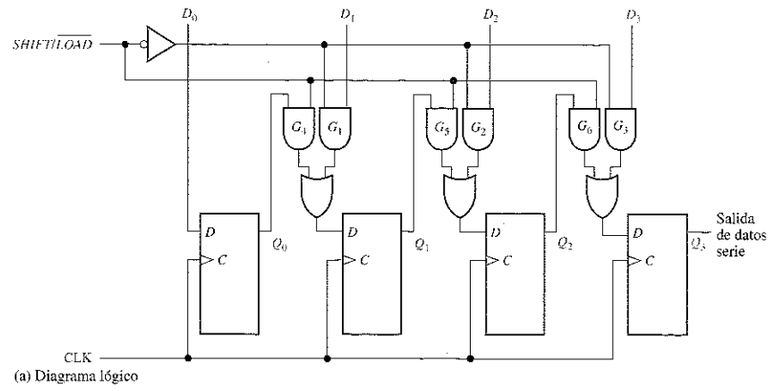
En un registro con entradas de datos paralelo, los bits se introducen simultáneamente en sus respectivas etapas a través de líneas paralelo, en lugar de bit a bit a través de una única línea como ocurre con las entradas de datos serie. La salida serie se hace del mismo modo que se ha descrito en la Sección 10.2, una vez que todos los datos están almacenados en el registro.

Al finalizar esta sección, el lector deberá ser capaz de:

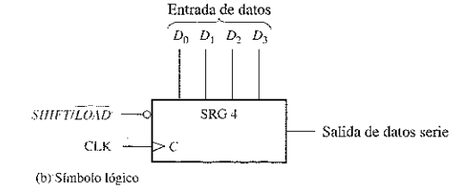
■ Explicar cómo se introducen los datos en el registro de desplazamiento con entrada paralelo. ■ Comparar las entradas serie y paralelo. ■ Utilizar el registro de desplazamiento de 8 bits de carga paralelo 74HC165. ■ Desarrollar y analizar los diagramas de tiempos de los registros con entrada paralelo / salida serie.

La Figura 10.12 ilustra un registro de desplazamiento de 4 bits con entrada paralelo y salida serie y su símbolo lógico. Obsérvese que tiene cuatro líneas de entrada de datos D_0, D_1, D_2 y D_3 y una entrada *SHIFT/$\overline{LOAD}$* (desplazamiento/carga), que permite **cargar** en paralelo los cuatro bits de datos en el registro. Cuando esta entrada está a nivel BAJO, las puertas G_1 a G_3 se activan, permitiendo que cada bit sea aplicado a la entrada D de su respectivo flip-flop. Cuando se aplica un impulso de reloj, los flip-flops con $D = 1$ pasan al estado SET, y los flip-flops con $D = 0$ pasan al estado RESET, almacenándose de este modo los cuatro bits simultáneamente.

Cuando la entrada *SHIFT/$\overline{LOAD}$* está a nivel ALTO, las puertas G_1 a G_3 se inhiben y las puertas G_4 a G_6 se activan, permitiendo que los bits de datos se desplacen hacia la derecha, pasando de una etapa a la siguiente. Las puertas OR permiten el desplazamiento normal o la introducción de datos en paralelo, dependiendo de qué puertas AND se hayan activado según el nivel de la entrada *SHIFT/$\overline{LOAD}$*.



(a) Diagrama lógico

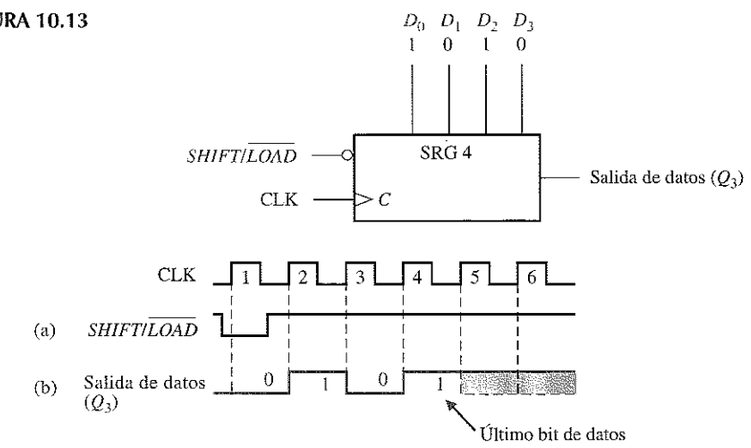


(b) Símbolo lógico

FIGURA 10.12
Registro de desplazamiento de 4 bits con entrada paralelo / salida serie.

EJEMPLO 10.3 Determinar la forma de onda de la salida de datos de un registro de 4 bits para las formas de onda de entrada paralelo de datos, de reloj y *SHIFT/LOAD* de la Figura 10.13(a). Utilizar el diagrama lógico de la Figura 10.12(a).

FIGURA 10.13



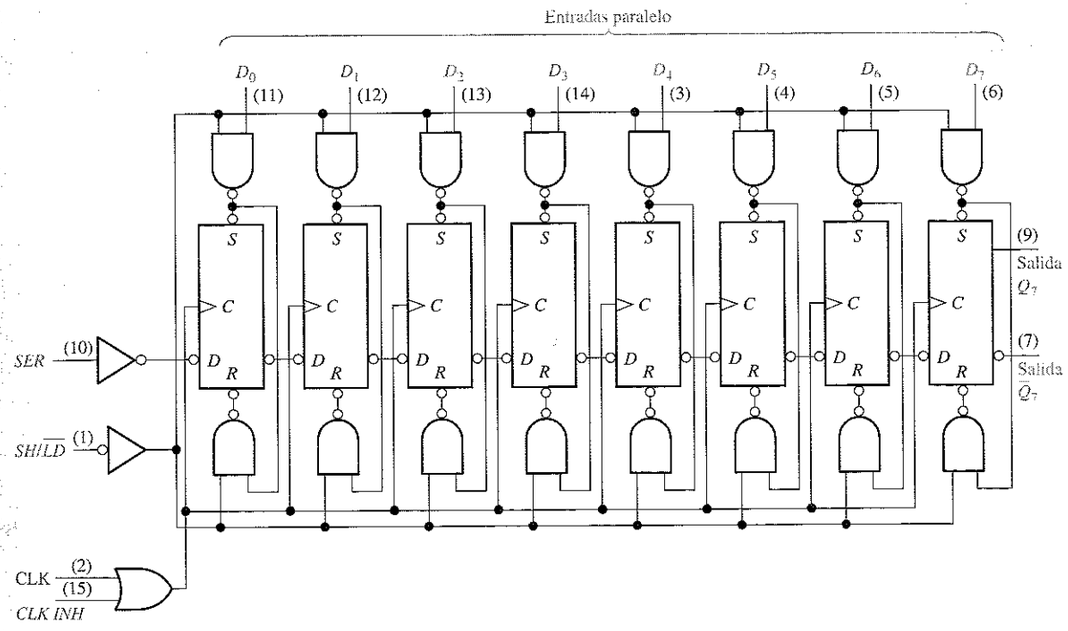
Solución. En el impulso de reloj 1, los datos paralelo ($D_0D_1D_2D_3 = 1010$) se cargan en el registro, poniendo la salida Q_3 a 0. En el impulso de reloj 2, el 1 de Q_2 se desplaza a Q_3; en el impulso de reloj 3, el 0 se desplaza a Q_3; en el impulso de reloj 4, el último bit de datos (1) se desplaza a Q_3 y en el impulso de reloj 5 todos los bits se han desplazado y salido del registro, y sólo quedan 1s en el mismo (suponiendo que la entrada D_0 permanece a 1). Véase la Figura 10.13(b).

Problema relacionado. Determinar la señal de salida de datos para las señales de reloj y *SHIFT/LOAD* de la Figura 10.13(a), si las entradas de datos paralelo son $D_0D_1D_2D_3 = 0101$.

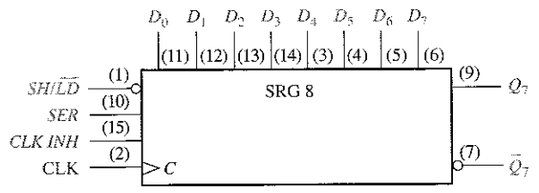
Un registro de desplazamiento de 8 bits con carga paralelo

El 74HC165 es un ejemplo de registro de desplazamiento en formato de circuito integrado que trabaja con entrada paralelo y salida serie (también puede funcionar con entrada y salida serie). La Figura 10.14(a) muestra el diagrama lógico interno de este dispositivo y en la parte (b) se presenta su símbolo lógico.

La entrada *SHIFT/LOAD (SH/LD)* a nivel BAJO activa todas las puertas NAND para realizar la carga paralelo. Cuando un bit de datos de entrada es un 1, el flip-flop pasa al estado SET de forma asíncrona debido al nivel BAJO en la salida de la puerta superior. Además, las datos se pueden introducir en serie a través de la entrada *SER*. El reloj se puede inhibir en cualquier instante aplicando un nivel ALTO a la entrada *CLK INH*. Las salidas de datos serie del registro son Q_7 y su complemento $\overline{Q}_7$. Esta implementación es distinta de la que se ha visto anteriormente, el método síncrono de carga paralelo, lo que demuestra que existen varias formas de realizar la misma función.



(a) Diagrama lógico



(b) Símbolo lógico

FIGURA 10.14
El registro de desplazamiento de 8 bits con carga paralelo 74HC165.

La Figura 10.15 es un cronograma que muestra un ejemplo de funcionamiento de un registro de desplazamiento 74HC165.

REVISIÓN DE LA SECCIÓN 10.4

1. Explicar la función de la entrada *SHIFT/LOAD*.
2. En un registro de desplazamiento 74HC165, la operación de carga paralelo ¿es síncrona o asíncrona? ¿qué quiere decir esto?

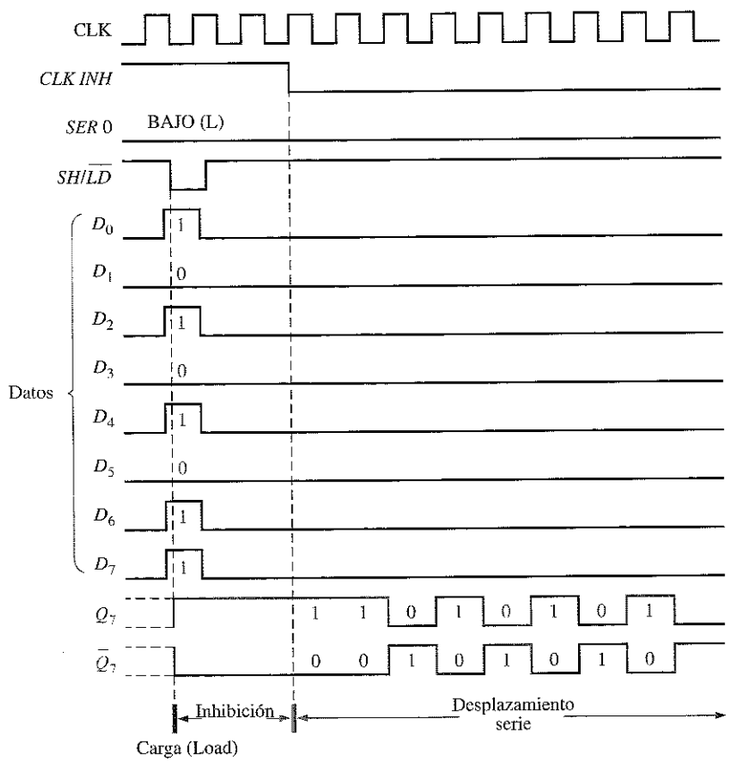


FIGURA 10.15
Cronograma de un registro de desplazamiento 74HC165.

10.5 ■ REGISTROS DE DESPLAZAMIENTO CON ENTRADA / SALIDA PARALELO

En la Sección 10.4 se ha descrito la entrada en paralelo de datos y la salida en paralelo de datos también se ha visto anteriormente. El registro de entrada y salida paralelo aplica ambos métodos. Inmediatamente después de introducir simultáneamente todos los bits de datos, éstos aparecen en paralelo en las salidas paralelo.

Al finalizar esta sección, el lector deberá ser capaz de:

- Utilizar el registro de desplazamiento de 4 bits de acceso paralelo 74HC195.
- Desarrollar y analizar el diagrama de tiempos para los registros de entrada / salida paralelo.

La Figura 10.16 presenta un registro de entrada / salida paralelo.

Un registro de desplazamiento de 4 bits de acceso paralelo

El 74HC195 puede utilizarse para trabajar con entrada y salida paralelo. Dado que también dispone de una entrada serie, se puede emplear para trabajar con entrada y salida serie, o entrada serie y salida paralelo. Puede usarse para funcionar con entrada

paralelo y salida serie utilizando Q_3 como salida. En la Figura 10.17 se muestra su símbolo lógico típico.

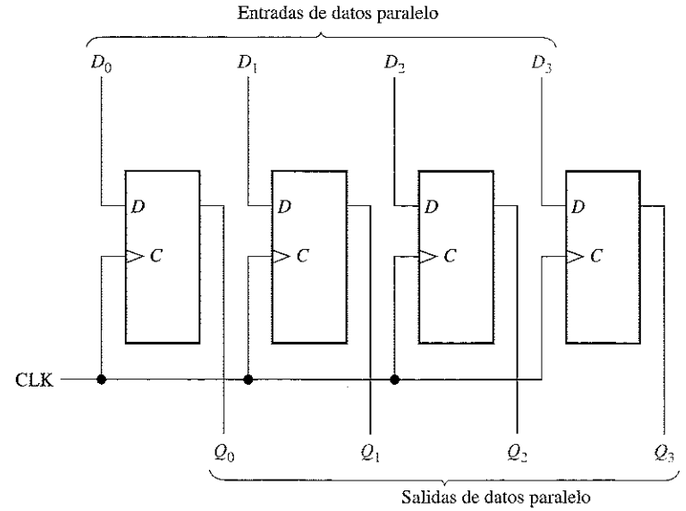


FIGURA 10.16
Registro de entrada / salida paralelo.

Cuando la entrada *SHIFT/$\overline{LOAD}$* (*SH/$\overline{LD}$*) está a nivel BAJO, los datos de las entradas paralelo se introducen en forma síncrona durante la transición positiva de la señal de reloj. Cuando *SH/$\overline{LD}$* está a nivel ALTO, los datos almacenados se desplazan a la derecha (Q_0 a Q_3), sincronizados con la señal de reloj. Las entradas J y $\overline{K}$ son para las entradas de datos serie de la primera etapa del registro (Q_0); Q_3 puede utilizarse como salida de datos serie. La entrada de borrado activa a nivel BAJO es asíncrona.

El diagrama de tiempos de la Figura 10.18 ilustra el funcionamiento de este registro.

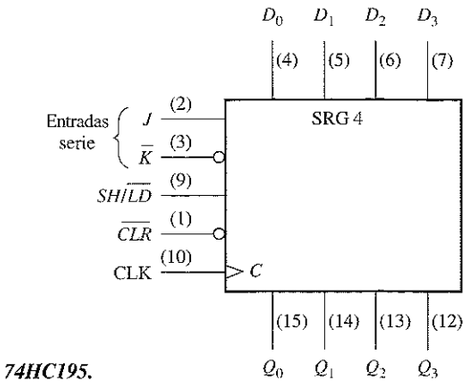


FIGURA 10.17
El registro de desplazamiento de 4 bits con acceso paralelo 74HC195.

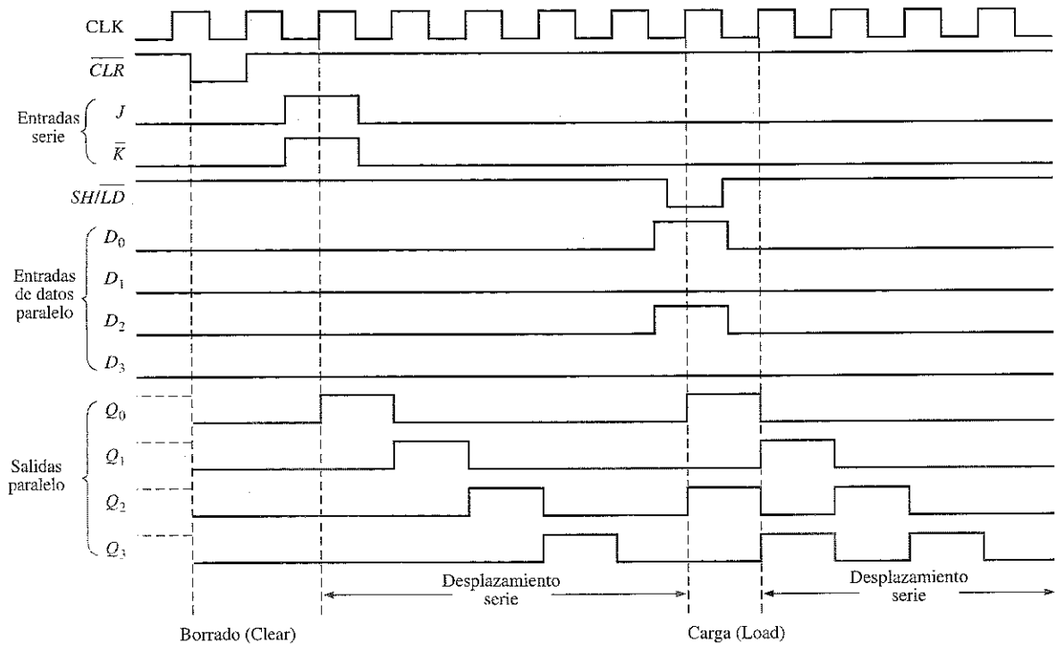


FIGURA 10.18
Diagrama de tiempos para un registro de desplazamiento 74HC195.

REVISIÓN DE LA SECCIÓN 10.5

1. En la Figura 10.16, $D_0 = 1$, $D_1 = 0$, $D_2 = 0$ y $D_3 = 1$. Después de tres impulsos de reloj, ¿cuáles son las salidas de datos?
2. Para un 74HC195, $SH/\overline{LD} = 1$, $J = 1$ y $\overline{K} = 1$. Después de un impulso de reloj, ¿cuál es el estado de Q_0?

10.6 ■ REGISTROS DE DESPLAZAMIENTO BIDIRECCIONALES

Un registro de desplazamiento bidireccional es aquel en el que los datos se pueden desplazar a izquierda o a derecha. Se puede implementar utilizando puertas lógicas que permitan la transferencia de un bit de datos de una etapa a la siguiente de la izquierda o de la derecha, dependiendo del nivel de una línea de control.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar el funcionamiento de un registro de desplazamiento bidireccional. ■ Utilizar el registro universal de desplazamiento bidireccional de 4 bits 74HC194. ■ Desarrollar y analizar los diagramas de tiempos de los registros de desplazamiento bidireccionales.

En la Figura 10.19 se muestra un registro de desplazamiento **bidireccional**. Un nivel ALTO en la entrada de control *RIGHT/$\overline{LEFT}$ (derecha/izquierda)* permite a los bits de

datos que están dentro del registro desplazarse hacia la derecha, y un nivel BAJO hace que se desplacen hacia la izquierda. Un examen de la lógica de puertas hará evidente este funcionamiento. Cuando la entrada de control *RIGHT/$\overline{LEFT}$* está a nivel ALTO, las puertas G_1 a G_4 se activan, y el estado de la salida Q de cada flip-flop pasa a la entrada D del *siguiente* flip-flop. Cuando se produce un impulso de reloj, los bits de datos se desplazan una posición a la *derecha*. Cuando esta entrada de control está a nivel BAJO, las puertas G_5 a G_8 se activan, y la salida Q de cada flip-flop pasa a la entrada D del flip-flop *precedente*. Cuando se genera un impulso de reloj, los bits de datos se desplazan una posición hacia la *izquierda*.

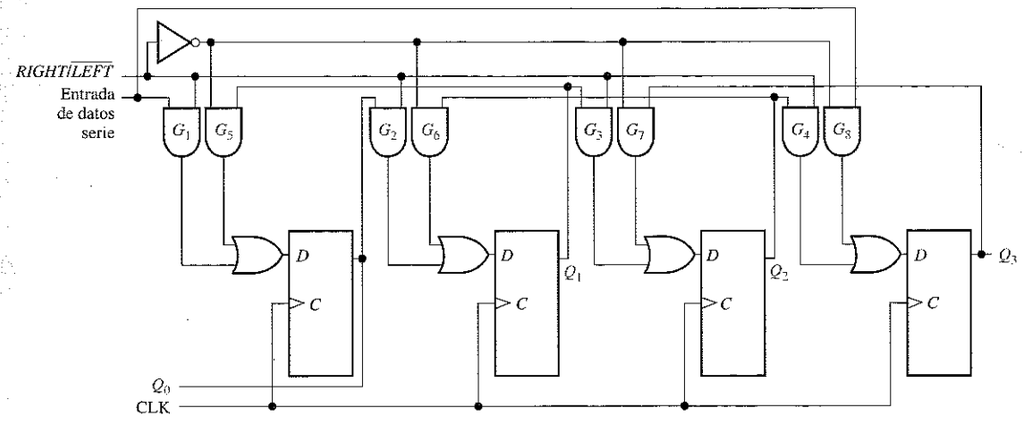


FIGURA 10.19
Registro de desplazamiento bidireccional de 4 bits.

EJEMPLO 10.4 Determinar el estado del registro de desplazamiento de la Figura 10.19 después de cada impulso de reloj para la forma de onda de la entrada de control *RIGHT/$\overline{LEFT}$* indicada en la Figura 10.20(a). Suponer que $Q_0 = 1$, $Q_1 = 1$, $Q_2 = 0$ y $Q_3 = 1$, y que la línea de entrada de datos serie está a nivel BAJO.

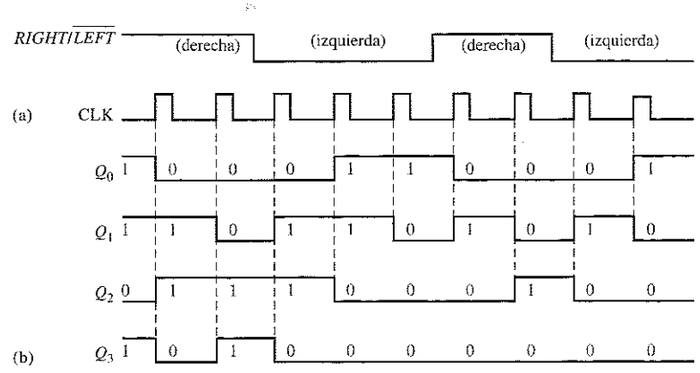


FIGURA 10.20

Solución. Véase la Figura 10.20(b).

Problema relacionado. Invertir la señal *RIGHT/LEFT* y determinar el estado del registro de desplazamiento de la Figura 10.19 después de cada impulso de reloj.

Un registro de desplazamiento universal bidireccional de 4 bits

El 74HC194 es un ejemplo de un registro de desplazamiento bidireccional universal en formato de circuito integrado. Un **registro de desplazamiento universal** tiene capacidad de entrada y salida serie y paralelo. En la Figura 10.21 se muestra su símbolo lógico y en la Figura 10.22 se presenta un ejemplo de diagrama de tiempos.

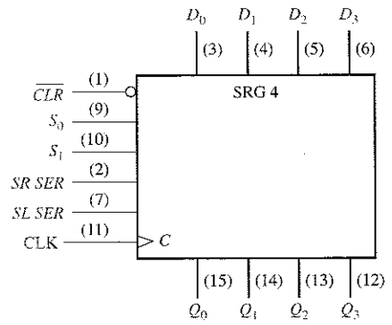


FIGURA 10.21
El registro de desplazamiento universal bidireccional de 4 bits 74HC194.

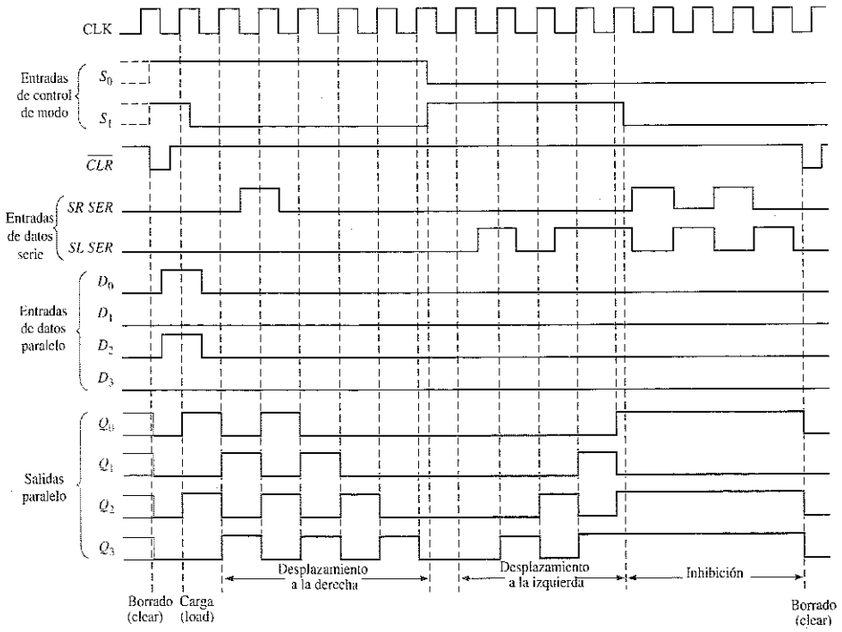


FIGURA 10.22
Cronograma de un registro de desplazamiento 74HC194.

La carga paralelo, que se sincroniza con una transición positiva de la señal de reloj, se consigue aplicando los cuatro bits de datos en las entradas paralelo y un nivel ALTO en las entradas S_0 y S_1. El desplazamiento a la derecha se consigue de forma síncrona con el flanco positivo del impulso de reloj cuando S_0 está a nivel ALTO y S_1 a nivel BAJO. En este modo, los datos serie se introducen por la entrada serie de desplazamiento a la derecha (*SR SER*). Cuando S_0 está a nivel BAJO y S_1 a nivel ALTO, los bits de datos se desplazan hacia la izquierda sincronizados con la señal de reloj, introduciendo nuevos datos por la entrada serie de desplazamiento a la izquierda (*SL SER*). La entrada *SR SER* entra en la etapa Q_0 y *SL SER* entra en la etapa Q_3.

REVISIÓN DE LA SECCIÓN 10.6

1. Suponer que el registro de desplazamiento bidireccional de 4 bits de la Figura 10.19 contiene: $Q_0 = 1$, $Q_1 = 1$, $Q_2 = 0$, $Q_3 = 0$. En la línea de entrada de datos serie hay un 1. Si la entrada *RIGHT/LEFT* está a nivel ALTO durante tres impulsos de reloj y a nivel BAJO para otros dos, ¿cuál será el contenido del registro después de los cinco impulsos de reloj?

10.7 ■ CONTADORES BASADOS EN REGISTRO DE DESPLAZAMIENTO

Un contador basado en un registro de desplazamiento es básicamente un registro de desplazamiento con la salida serie realimentada a la entrada serie, de modo que se generen secuencias especiales. A menudo estos dispositivos se clasifican como contadores porque disponen de una secuencia de estados específica. En esta sección, se introducen dos de los tipos más comunes de esta clase de contadores, el contador Johnson y el contador en anillo.

Al finalizar esta sección, el lector deberá ser capaz de:

- Establecer en qué se diferencia un contador basado en un registro de desplazamiento de un registro de desplazamiento básico.
- Explicar el funcionamiento de un contador Johnson.
- Especificar una secuencia de Johnson para cualquier número de bits.
- Explicar el funcionamiento de un contador en anillo y determinar la secuencia de cualquier contador en anillo específico.

El contador Johnson

En un **contador Johnson** el complemento de la salida del último flip-flop se conecta a la entrada *D* del primer flip-flop (también se puede implementar con otros tipos de flip-flop). Esta realimentación permite generar una secuencia de estados característica, tal y como muestran las Tablas 10.1 y 10.2 para un dispositivo de 4 bits y otro de 5 bits, respectivamente. Obsérvese que la secuencia de 4 bits tiene un total de ocho estados, o patrones de bits, y que la secuencia de 5 bits establece un total de diez estados. En general, un contador Johnson generará un módulo de $2n$, donde n es el número de etapas del contador.

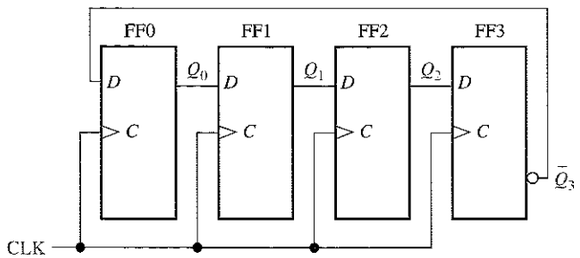
En la Figura 10.23 se muestra la implementación de los contadores Johnson de 4 y 5 etapas.

TABLA 10.1
Secuencia Johnson de cuatro bits

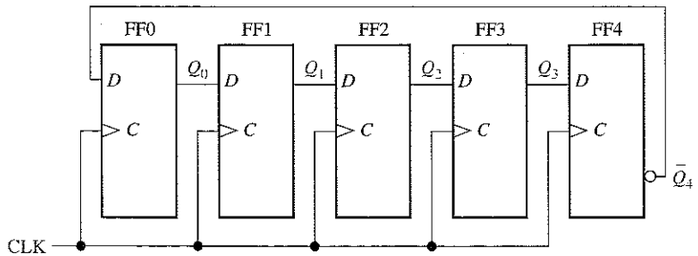
Pulso de reloj	Q_0	Q_1	Q_2	Q_3
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1

TABLA 10.2
Secuencia Johnson de cinco bits.

Pulso de reloj	Q_0	Q_1	Q_2	Q_3	Q_4
0	0	0	0	0	0
1	1	0	0	0	0
2	1	1	0	0	0
3	1	1	1	0	0
4	1	1	1	1	0
5	1	1	1	1	1
6	0	1	1	1	1
7	0	0	1	1	1
8	0	0	0	1	1
9	0	0	0	0	1



(a) Contador Johnson de cuatro-bits.



(b) Contador Johnson de cinco-bits.

FIGURA 10.23
Contadores Johnson de cuatro y cinco bits.

La implementación de un contador Johnson es muy sencilla e independiente del número de etapas. La salida Q de cada etapa se conecta a la entrada D de la etapa siguiente (suponiendo que se utilizan flip-flops D). La única excepción es que la salida $\overline{Q}$ de la última etapa se conecta a la entrada D de la primera etapa. Como indican las secuencias de las Tablas 10.1 y 10.2, el contador se "llenará" de 1s de izquierda a derecha, y luego se "llenará" de nuevo de 0s.

En las Figuras 10.24 y 10.25 se muestran, respectivamente, los diagramas de tiempos de los contadores de 4 y 5 bits.

El contador en anillo

El **contador en anillo** utiliza un flip-flop para cada estado de su secuencia. Tiene la ventaja de que no se requieren puertas de decodificación. En el caso de un contador en anillo de 10 bits, hay una única salida para cada dígito decimal.

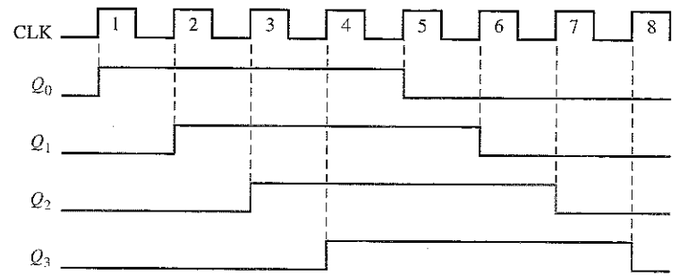


FIGURA 10.24
Secuencia de temporización del contador Johnson de 4 bits.

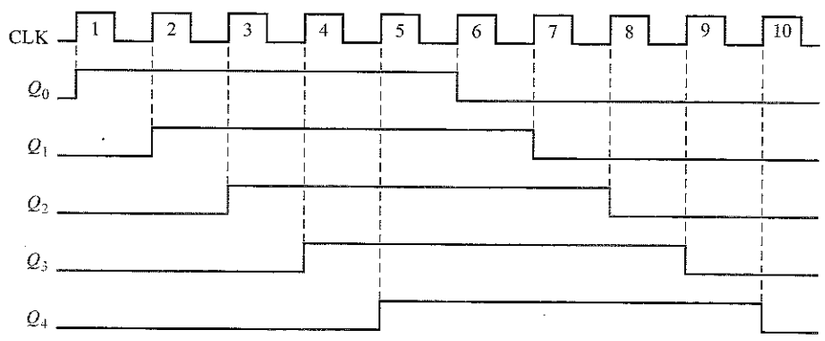


FIGURA 10.25
Secuencia de temporización del contador Johnson de 5 bits.

En la Figura 10.26 se muestra un diagrama lógico para un contador en anillo de 10 bits. En la Tabla 10.3 se facilita la secuencia de este mismo contador. Inicialmente, se presenta un 1 en el primer flip-flop, y se borran los restantes flip-flops. Obsérvese que las conexiones entre etapas son iguales a las del contador Johnson, excepto que en este caso es la salida Q de la última etapa, en lugar de $\overline{Q}$, la que se realimenta. Las diez salidas del contador indican directamente el valor decimal de la cuenta de los impulsos de reloj. Por ejemplo, un 1 en Q_0 representa un cero; un 1 en Q_1 indica uno; un 1 en Q_2 corresponde a dos en decimal; un 1 en Q_3 corresponde a tres, y así sucesiva-

mente. Debería verificar uno mismo que sólo un 1 se mantiene en el contador y que éste simplemente se desplaza "alrededor del anillo", avanzando una etapa con cada impulso de reloj.

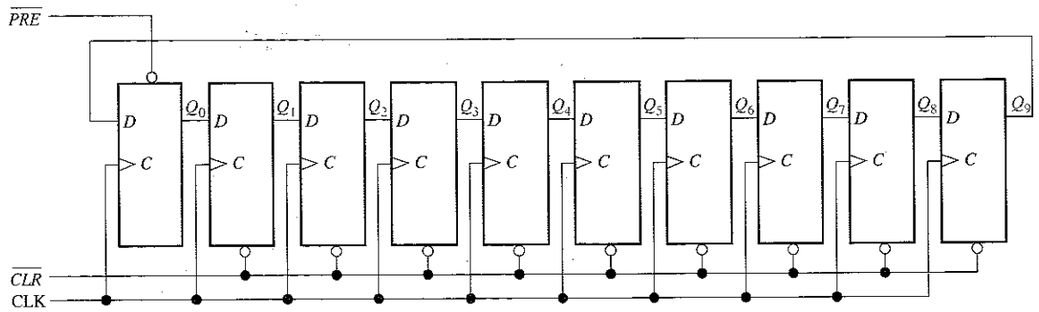


FIGURA 10.26
Contador en anillo de diez bits.

Como se ilustra en el Ejemplo 10.5, se pueden conseguir otras secuencias introduciendo más de un 1 en el contador.

TABLA 10.3
Secuencia del contador en anillo de diez bits.

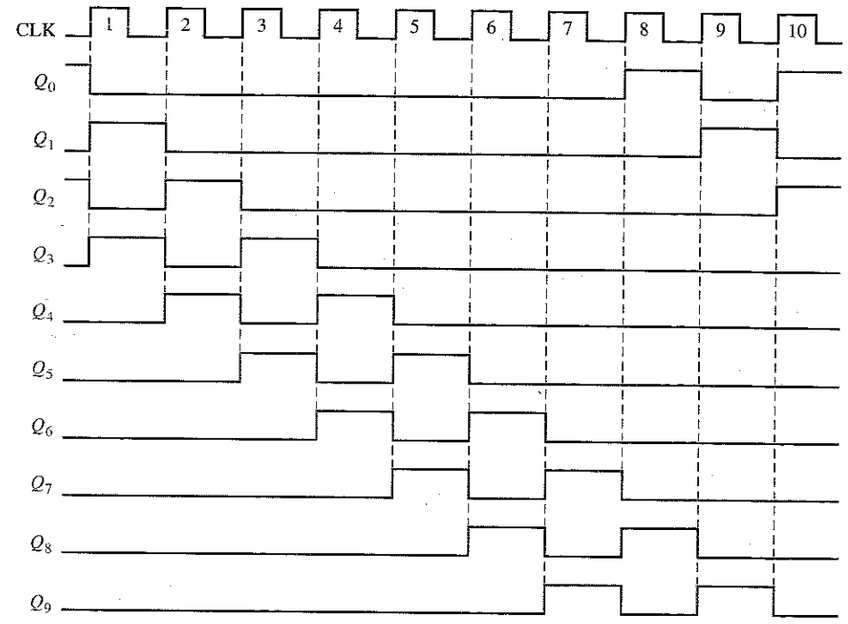
Pulso de reloj	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9
0	1	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0
5	0	0	0	0	0	1	0	0	0	0
6	0	0	0	0	0	0	1	0	0	0
7	0	0	0	0	0	0	0	1	0	0
8	0	0	0	0	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	1

EJEMPLO 10.5 Si el contador en anillo de 10 bits de la Figura 10.26 tiene el estado inicial 1010000000, determinar la forma de onda para cada una de las salidas Q.

Solución. Véase la Figura 10.27.

Problema relacionado. Si un contador en anillo de 10 bits está en el estado inicial 0101001111, determinar la forma de onda de cada salida Q.

FIGURA 10.27



REVISIÓN DE LA SECCIÓN 10.7

1. ¿Cuántos estados tiene un contador Johnson de 8 bits?
2. Escribir la secuencia de estados de un contador Johnson de 3 bits que se inicia en el estado 000.

10.8 ■ APLICACIONES DE LOS REGISTROS DE DESPLAZAMIENTO

NOTAS INFORMÁTICAS
Los registros de propósito general en el Pentium son registros de 32 bits que se pueden utilizar para el almacenamiento temporal de datos, así como para usos específicos.

Los registros de desplazamiento se encuentran en muchos tipos de aplicaciones; en esta sección, se van a ver algunas de ellas.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Utilizar un registro de desplazamiento para generar un retardo de tiempo. ■ Implementar una secuencia específica de un contador en anillo utilizando un registro de desplazamiento 74HC195. ■ Explicar cómo se usan los registros de desplazamiento para la conversión de datos serie/paralelo. ■ Definir *UART*. ■ Explicar el funcionamiento de un codificador de teclado y cómo se utilizan los registros en esta aplicación.

Retardo de tiempo

Los registros de desplazamiento con entrada y salida serie se usan para obtener un retardo de tiempo de la entrada a la salida, que es función del número de etapas (n) del registro y de la frecuencia de reloj.

Cuatro de estos registros son los siguientes: el *acumulador* (EAX), que se utiliza principalmente para el almacenamiento temporal de datos y de operandos de instrucciones; el *registro base* (EBX), que se utiliza para almacenar un valor de forma temporal; el *registro contador* (ECX), que se usa principalmente para determinar el número de repeticiones en bucles, operaciones con cadenas de caracteres, desplazamientos o rotaciones. El *registro de datos* (EDX), que normalmente se emplea para el almacenamiento temporal de datos.

Cuando se aplica un impulso de datos a la entrada serie de la Figura 10.28 (*A* y *B* se conectan juntas), éste se introduce en la primera etapa sincronizado con el flanco de disparo del impulso de reloj. El dato se desplaza de etapa en etapa con cada impulso de reloj sucesivo hasta que aparece en la salida serie *n* periodos de reloj más tarde. En la Figura 10.28 se ilustra este funcionamiento, utilizando un registro de desplazamiento con entrada y salida serie de 8 bits y una frecuencia de reloj de 1 MHz, para conseguir un retardo de tiempo (t_d) de 8 μs (8 x 1 μs). Este retardo se puede aumentar o disminuir variando la frecuencia de reloj. El retardo de tiempo también se puede incrementar conectando en cascada registros de desplazamiento, y se puede decrementar tomando sucesivamente la salida de las etapas intermedias del registro, si están disponibles, como ilustra el Ejemplo 10.6.

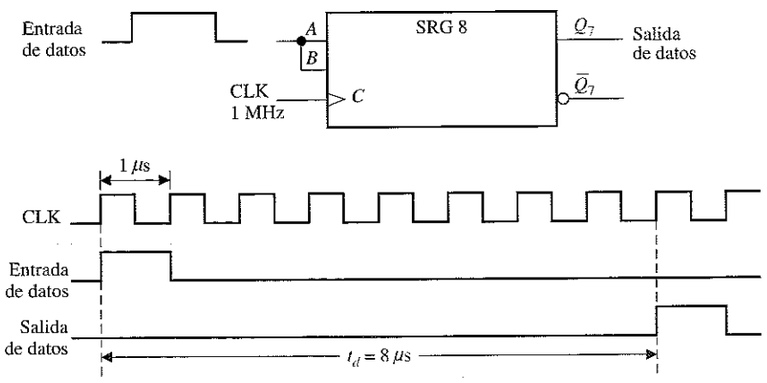
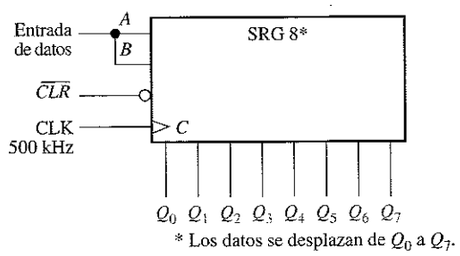


FIGURA 10.28
El registro de desplazamiento como dispositivo de retardo de tiempo.

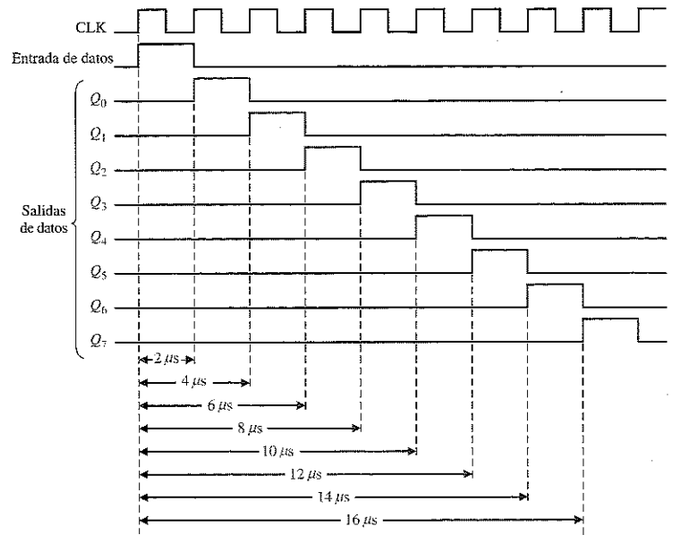
EJEMPLO 10.6 Determinar el retardo de tiempo entre la entrada serie y cada salida del registro de la Figura 10.29. Realizar un diagrama de tiempos para ilustrarlo.

FIGURA 10.29



Solución. El periodo de reloj es 2 μs. Luego el retardo de tiempo puede incrementarse o decrementarse de dos en dos μs, desde un mínimo de 2 μs hasta un máximo de 16 μs, como ilustra la Figura 10.30.

FIGURA 10.30
Diagrama de tiempos que muestra los retardos para el registro de la Figura 10.29.



Problema relacionado. Determinar la frecuencia de reloj requerida para obtener un retardo de tiempo de 24 μs en la salida Q_7 del registro de la Figura 10.29.

Contador en anillo que utiliza un registro de desplazamiento 74HC195

Un registro de desplazamiento se puede utilizar como contador en anillo si la salida se realimenta a la entrada serie. La Figura 10.31 ilustra esta aplicación empleando un registro de desplazamiento de 4 bits 74HC195.

Inicialmente, la secuencia patrón de bits 1000 (o cualquier otra) se introduce en el contador en modo síncrono para inicializarlo, aplicando dicha secuencia patrón a las entradas de datos paralelo, con la entrada *SH/$\overline{LD}$* a nivel BAJO y aplicando un impulso de reloj. Después de esta inicialización, el 1 se desplaza a través del contador en anillo, tal y como muestra el diagrama de tiempos de la Figura 10.32.

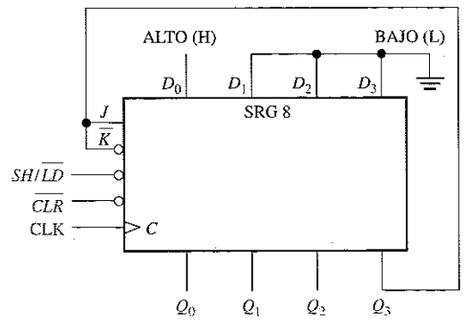


FIGURA 10.31
Un 74HC195 conectado como contador en anillo.

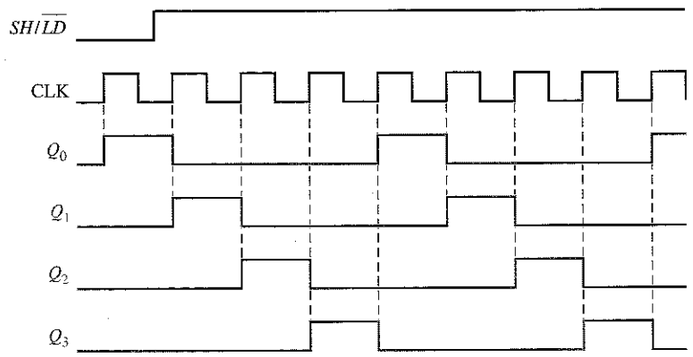


FIGURA 10.32
Diagrama de tiempos que muestra dos ciclos completos del contador en anillo de la Figura 10.31, cuando se inicializa con el estado 1000.

Convertidor de datos serie-paralelo

La transmisión de datos serie de un sistema digital a otro se usa comúnmente para reducir el número de conductores de la línea de transmisión. Por ejemplo, se pueden enviar en serie ocho bits por un único conductor, los cuales precisarían ocho conductores para transmitirse en paralelo.

Una computadora o un sistema basado en microprocesador, normalmente, requiere que la entrada de datos se haga en paralelo, por lo que es preciso realizar una conversión serie-paralelo. En la Figura 10.33 se muestra un convertidor de datos serie-paralelo simplificado, en el que se emplean dos tipos de registros de desplazamiento.

Para ilustrar el funcionamiento del convertidor serie-paralelo, utilizaremos el formato de datos serie de la Figura 10.34, compuesto de once bits. El primer bit (bit de arranque) siempre es 0 y siempre se inicia en una transición de nivel ALTO a nivel BAJO. Los siguientes ocho bits (D_7 a D_0) son los bits de datos (uno de los bits puede ser de paridad), y los dos últimos bits (bits de parada) son siempre 1. Cuando no se transmiten datos, la línea de datos serie siempre está a 1.

La transición de nivel ALTO a nivel BAJO del bit de arranque activa el flip-flop de control, el cual a su vez activa el generador de reloj. Después de un retardo fijo, el generador de reloj comienza a generar un tren de impulsos, que se aplica al registro de entrada de datos y al contador de división por 8. La señal de reloj tiene exactamente una frecuencia igual a la de los datos serie de entrada, y el primer impulso de reloj se produce después del bit de arranque y simultáneamente con el primer bit de datos.

El diagrama de tiempos de la Figura 10.35 ilustra la siguiente operación básica: los ocho bits de datos (D_7 a D_0) se desplazan en serie a través del registro de entrada de datos. Después del octavo impulso de reloj se produce una transición de nivel ALTO a nivel BAJO en la salida *TC* del contador a la que se aplica la operación AND con la señal de reloj (*TC*·CLK), lo que hace que los ocho bits se carguen en el registro de salida de datos. Esta misma transición también dispara el monoestable, el cual produce un impulso de corta duración que borra el contador, pone en estado de RESET el flip-flop de control y desactiva el generador de reloj. Ahora, el sistema está preparado

para recibir el siguiente grupo de once bits, y queda a la espera de que se produzca la siguiente transición de nivel ALTO a nivel BAJO del bit de arranque.

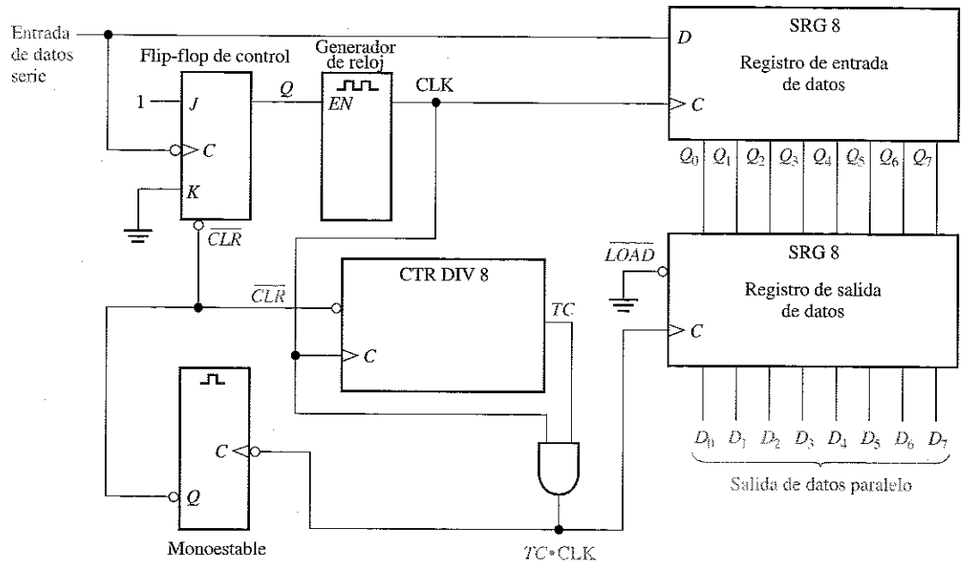


FIGURA 10.33
Diagrama lógico simplificado de un convertidor serie-paralelo.

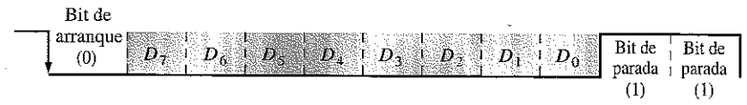


FIGURA 10.34
Formato de datos serie.

La conversión de datos paralelo-serie se puede realizar invirtiendo el proceso que se acaba de establecer. Sin embargo, puesto que se deben generar los datos en serie, es preciso considerar otros requisitos adicionales. En el problema 43 del final del capítulo se lleva a cabo el diseño de un convertidor de datos paralelo-serie de 8 bits.

Transmisor receptor asíncrono universal (UART, Universal Asynchronous Receiver Transmitter)

Como ya se ha mencionado, las computadoras y sistemas basados en microprocesador, a menudo, transmiten y reciben datos en paralelo. Frecuentemente, estos sistemas deben comunicarse con dispositivos externos que envían y/o reciben los datos en

serie. Un dispositivo que realiza la interfaz de conversión es el transmisor-receptor asíncrono universal (UART). En la Figura 10.36 se ilustra una UART en una aplicación general de un sistema basado en microprocesador.

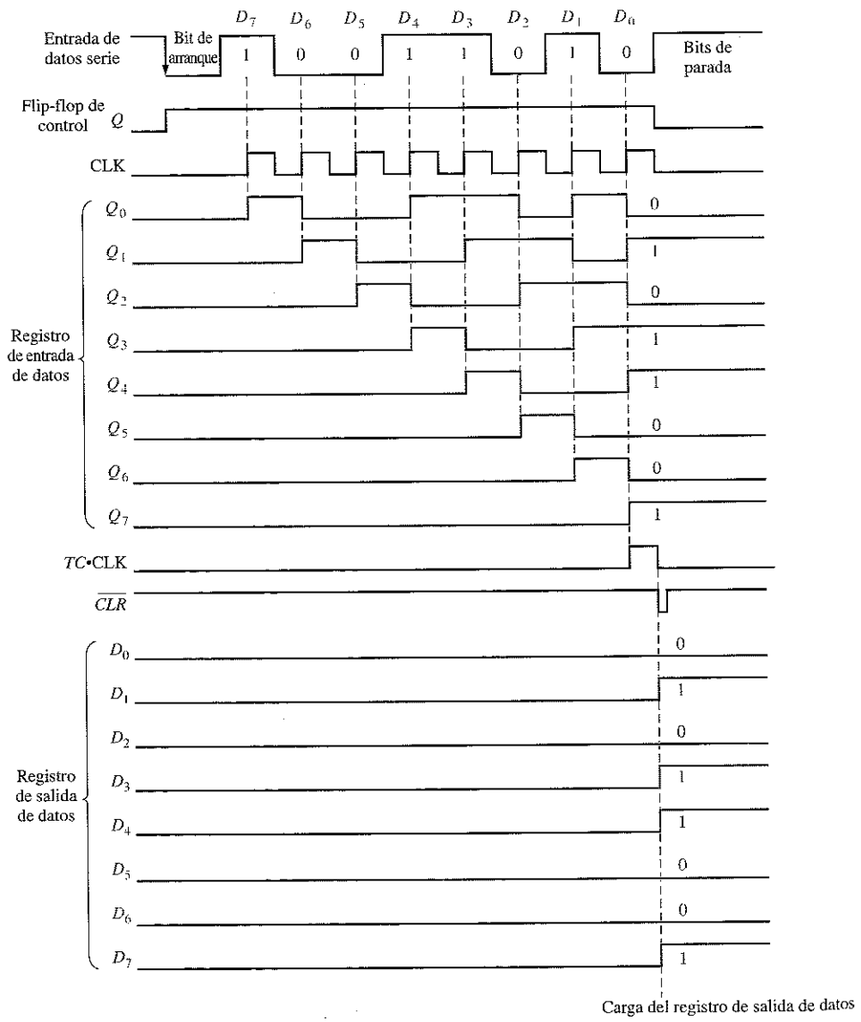


FIGURA 10.35
Diagrama de tiempos que ilustra el funcionamiento del convertidor de datos serie/paralelo de la Figura 10.33.

Una UART incluye un convertidor de datos serie/paralelo, como el que se ha explicado y un convertidor paralelo/serie, como muestra la Figura 10.37. El bus de datos es básicamente un conjunto de conductores paralelos a lo largo de los cuales los datos se

desplazan entre la UART y el microprocesador. Los buffers conectan los registros de datos con el bus de datos.

La UART recibe los datos en serie, los convierte a formato paralelo y los coloca en el bus de datos. La UART también acepta datos paralelo del bus de datos, los convierte a formato serie y los transmite al dispositivo externo.

FIGURA 10.36
Interfaz de una UART.

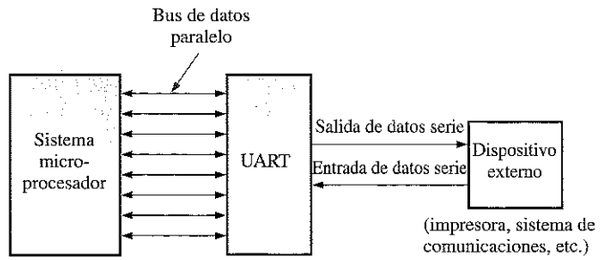
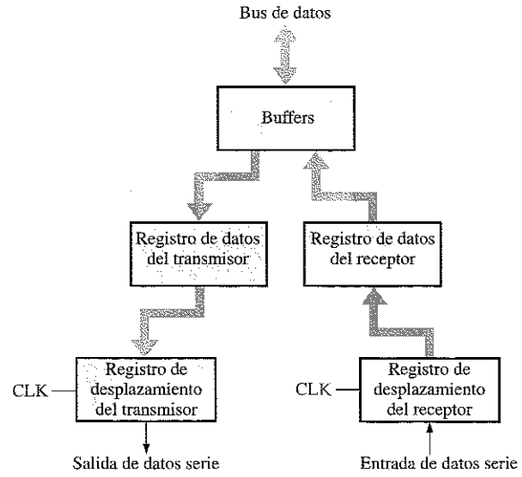


FIGURA 10.37
Diagrama de bloques básico de una UART.



Codificador de teclado

El codificador de teclado es un buen ejemplo de aplicación de un registro de desplazamiento utilizado como contador en anillo, junto con otros dispositivos. Recuérdese que, en el Capítulo 6, se presentó un codificador simplificado de teclado de computadora sin almacenamiento de datos.

La Figura 10.38 muestra un codificador de teclado simplificado que codifica la pulsación de una tecla en una matriz organizada en ocho filas y ocho columnas, que define 64 teclas. Se conectan dos registros de desplazamiento de 4 bits 74HC195 como contador en anillo de 8 bits con una secuencia patrón de bits formada por siete unos y un cero, que se activa cuando se conecta la alimentación. Se usan dos codificadores con prioridad 74HC147 (vistos en el Capítulo 6) como codificadores de ocho-líneas a tres-líneas (la entrada 9 a nivel ALTO, la salida 8 no se utiliza), para codificar las FILAS y las COLUMNAS de la matriz de teclado. El 74HC174A (séxtuple flip-

flop) se usa como registro con entrada y salida paralelo en el que se almacena el código FILA/COLUMNA procedente de los codificadores con prioridad.

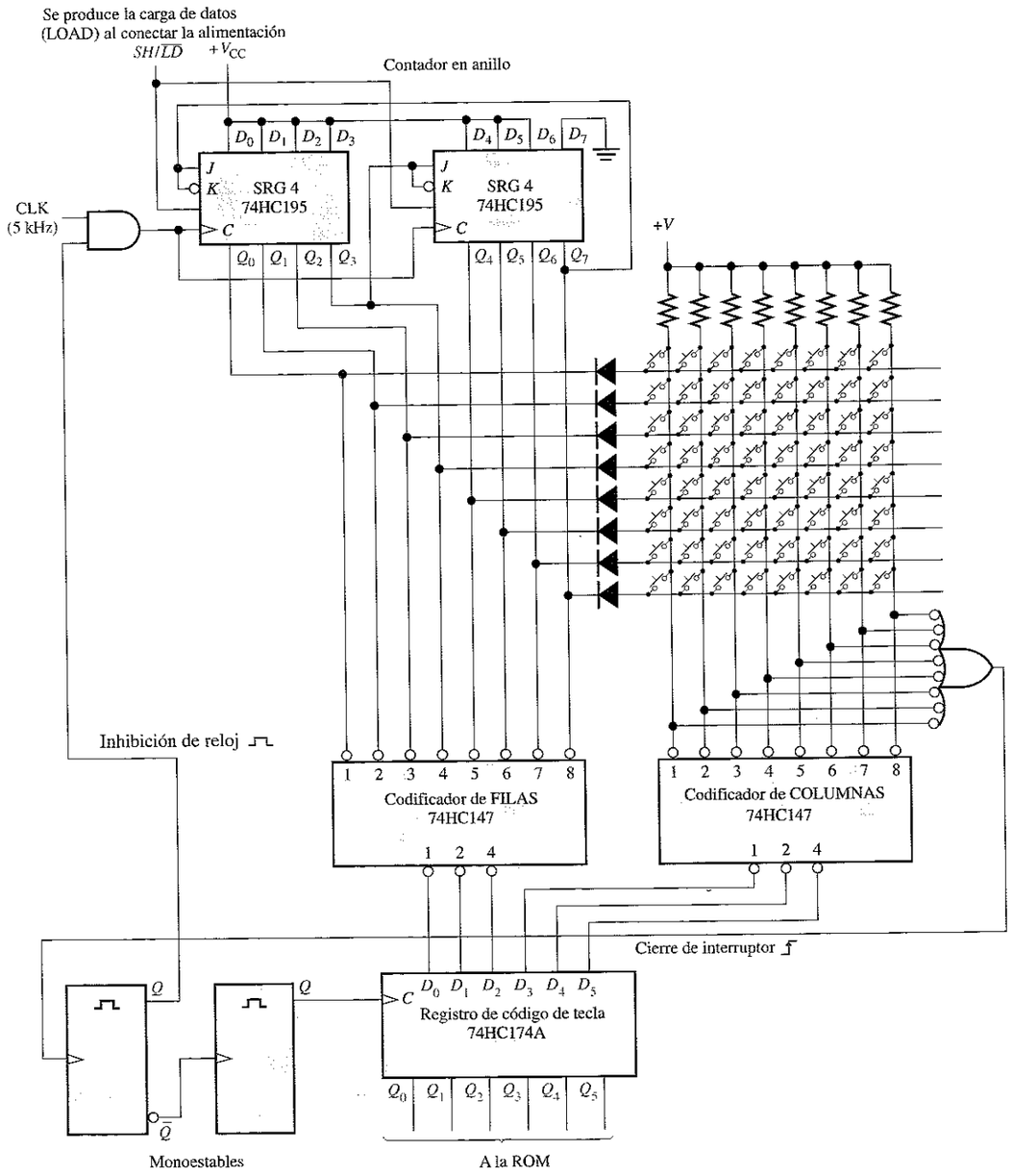


FIGURA 10.38
Circuito simplificado de codificación de teclado.

El funcionamiento básico del codificador de teclado de la Figura 10.38 es el siguiente: el contador en anillo "explora" las filas para detectar la pulsación de una tecla, a medida que la señal de reloj desplaza el 0 por el contador a una frecuencia de 5 kHz. Secuencialmente se aplica un 0 (nivel BAJO) a una de las líneas de FILA, mientras que las restantes líneas de FILA están a nivel ALTO. Todas las líneas FILA se conectan a las entradas del codificador de filas, de modo que la salida de 3 bits del mismo, en cualquier instante, representa, en binario, la línea FILA que está a nivel BAJO. Cuando se pulsa una tecla, la línea de COLUMNA correspondiente se conecta a la línea de FILA. Cuando el contador en anillo toma la línea de FILA que está a nivel BAJO, dicha COLUMNA también quedará a nivel BAJO. El codificador de columnas genera la salida binaria que corresponde a la COLUMNA en la que está la tecla pulsada. El código de tres bits de la FILA más el código de tres bits de la COLUMNA identifican unívocamente la tecla que se ha presionado. Este código de seis bits se aplica a las entradas del registro de código de tecla. Cuando se ha pulsado una tecla, los dos monoestables producen un impulso de reloj retrasado, para realizar la carga paralelo del código de seis bits en el registro de código de tecla. Este retraso permite que se extingan los rebotes de los contactos. La salida del primer monoestable también inhibe al contador en anillo, para evitar la exploración mientras que se están cargando los datos en el registro de códigos de las teclas.

Este código de 6 bits contenido en el registro de código de tecla se aplica ahora a una memoria ROM (read-only memory, memoria de sólo lectura) para convertirse en un código alfanumérico apropiado que identifique los caracteres del teclado. Las memorias ROM se estudian en el Capítulo 12.

REVISIÓN DE LA SECCIÓN 10.8

1. En el codificador de teclado, ¿cuántas veces por segundo explora el contador en anillo el teclado?
2. ¿Cuál es el código de 6 bits FILA/COLUMNA (código de tecla) para la fila superior y la columna más a la izquierda del codificador de teclado?
3. ¿Cuál es el propósito de los diodos en el codificador de teclado? ¿Cuál es la finalidad de las resistencias?

10.9 ■ LOCALIZACIÓN DE AVERÍAS

En esta sección, vamos a ocuparnos de un método básico para la localización de averías en sistemas lógicos secuenciales y otros sistemas digitales más complejos. Dicho método se basa en ejercitar (probar) al circuito bajo prueba con una señal de entrada conocida (estímulo), y luego observar la salida para la secuencia patrón de bits.

Al completar esta sección, el lector deberá ser capaz de:

- Explicar el procedimiento de "ejercitar" un circuito como técnica para la localización de averías.
- Aplicar el procedimiento de "ejercitar" a un convertidor serie-paralelo.
- Establecer los fundamentos del análisis de signatura.

El convertidor de datos serie-paralelo de la Figura 10.33 se usa para ilustrar el procedimiento de "ejercitar" un circuito. El objetivo principal de este procedimiento es forzar a todos los elementos del circuito (flip-flop y puertas) a que pasen por todos sus estados, con el fin de estar seguros de que en ningún estado determinado se produce un fallo. La secuencia patrón de prueba de entrada, en este caso, debe diseñarse para forzar a cada flip-flop de los registros a pasar por ambos estados, hacer que el contador pase por los ocho estados, y comprobar el flip-flop de control, el generador de reloj, el monoestable y la puerta AND.

La secuencia patrón de prueba de entrada que cumple este objetivo para el convertidor de datos serie-paralelo está basada en el formato de datos serie de la Figura 10.34. Se forma mediante el grupo serie de bits de datos 10101010 seguido de otro grupo 01010101, como muestra la Figura 10.39. Estas secuencias patrón se generan de forma repetitiva a partir de un generador especial de secuencias de prueba. En la Figura 10.40 se muestra la configuración básica de prueba.

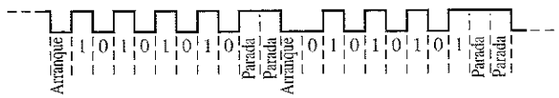


FIGURA 10.39
Secuencia patrón de prueba.

Después de que ambas secuencias patrón han pasado por el circuito bajo prueba, todos los flip-flops de los registros de entrada y salida de datos han pasado por los estados SET y RESET, el contador ha pasado a través de su secuencia (una vez para cada secuencia patrón de bits) y todos los demás dispositivos se han comprobado.

Para comprobar el funcionamiento, hay que observar cada una de las salidas de datos paralelo, para las dos secuencias patrón alternativas de 1s y 0s, mientras que las secuencias patrón de prueba de entrada se desplazan repetitivamente por el registro de datos de entrada y luego se cargan en el registro de salida de datos. En la Figura 10.41 se presenta el diagrama de tiempos. Las salidas se pueden observar por parejas utilizando un osciloscopio de doble traza, o se pueden ver las ocho simultáneamente con un analizador lógico configurado para realizar análisis de tiempos.

Si una o más de las salidas del registro de salida de datos es incorrecta, se deben comprobar las salidas del registro de entrada de datos. Si estas salidas son correctas, entonces el problema estará asociado al registro de salida de datos. Comprobar las entradas al registro de salida de datos directamente sobre los pines del CI para detectar una línea en circuito abierto. Comprobar también que las conexiones de alimentación y tierra son correctas (buscando la ausencia de ruido en la línea de tierra). Verifique que la línea de carga está a nivel BAJO y que en la entrada de la señal de reloj hay impulsos de reloj de amplitud correcta. Asegurarse de que la conexión al analizador lógico no conecte dos líneas de salida, dando lugar a un cortocircuito. Si todas esta pruebas pasan la inspección, entonces, probablemente, es que el registro de salida sea defectuoso. Si las salidas del registro de entrada de datos son también incorrectas, el fallo podría estar en el propio registro de entrada o en cualquier otra parte de la lógica, por lo que será necesaria una investigación adicional para aislar el problema.

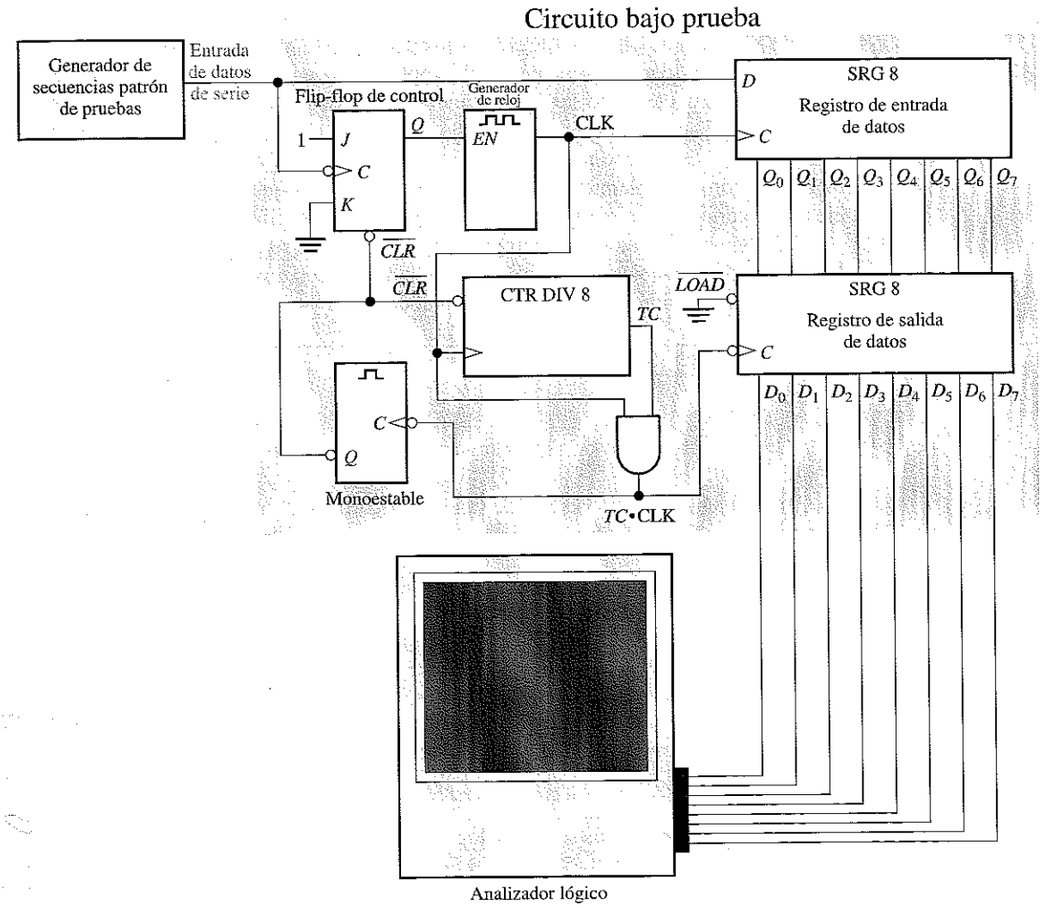


FIGURA 10.40
Configuración de prueba básica para el convertidor de datos serie-paralelo de la Figura 10.33.

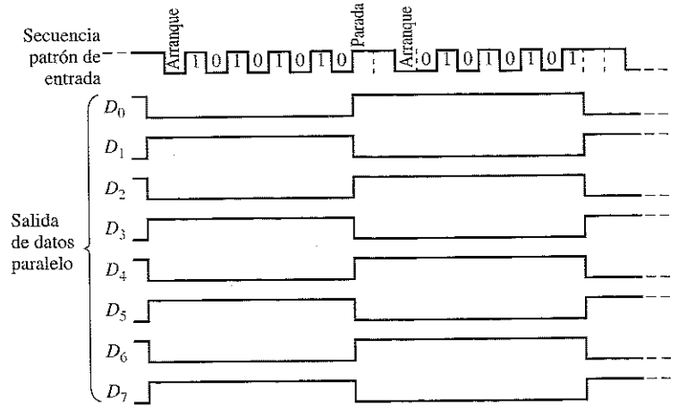


FIGURA 10.41
Salidas correctas del circuito bajo prueba de la Figura 10.40. Se muestran también la secuencia patrón de prueba de entrada.

Consejos Prácticos

Cuando se miden señales digitales con un osciloscopio, siempre se debería utilizar el acoplamiento en continua en lugar del acoplamiento en alterna. La razón de que el acoplamiento en alterna no sea mejor para visualizar señales digitales es que el nivel de 0 V de la señal aparecerá en el nivel medio de la señal, no en el verdadero nivel de tierra o nivel de 0 V. Es mucho más sencillo encontrar una tierra "flotante" o un nivel lógico incorrecto con el acoplamiento en continua. Si se sospecha que hay un punto de tierra en circuito abierto en un circuito digital, incrementar la sensibilidad del osciloscopio hasta el máximo posible. Una buena tierra nunca aparecerá con ruido bajo estas condiciones, aunque un circuito abierto probablemente se mostrará con algo de ruido, lo que aparece como una fluctuación aleatoria sobre el nivel de 0 V.

REVISIÓN DE LA SECCIÓN 10.9

1. ¿Cuál es el propósito de proporcionar una entrada de prueba a un circuito lógico secuencial?
2. Generalmente, cuando la señal de salida es incorrecta, ¿cuál es el siguiente paso que se debe dar?

10.10 ■ SÍMBOLOS LÓGICOS CON NOTACIÓN DE DEPENDENCIA

Esta sección puede omitirse, puesto que no afecta al resto de los temas. Se presentan dos ejemplos de símbolos con notación de dependencia, según el estándar ANSI/IEEE 91-1984, para los registros de desplazamiento.

Al finalizar esta sección, el lector deberá ser capaz de:

- Entender e interpretar los símbolos lógicos con notación de dependencia para los registros de desplazamiento 74HC164 y 74HC194.

En la Figura 10.42 se presenta el símbolo lógico de un registro de desplazamiento serie con salida paralelo de 8 bits 74HC164. Las entradas de control comunes se indican en el bloque. La entrada de borrado ($\overline{CLR}$) se indica con la letra *R* (RESET) en el interior del bloque. Puesto que no existe prefijo de dependencia para enlazar *R* con el reloj (C1), la función de borrado es asíncrona. La flecha a la derecha de C1 indica el flujo de datos de Q_0 a Q_7. A las entradas *A y B* se les aplica la operación AND, como indica el símbolo AND especificado en el interior del bloque, lo que proporciona la entrada de datos síncrona, 1*D*, en la primera etapa (Q_0). Obsérvese que la dependencia de *D* y *C* se indica mediante el sufijo 1 para *C* y el prefijo 1 para *D*.

La Figura 10.43 es el símbolo lógico para el registro de desplazamiento universal 74HC194. Empezando por la parte superior izquierda del bloque, vemos que la entrada $\overline{CLR}$ es activa a nivel BAJO y asíncrona (no hay prefijo de enlace con *C*). Las entradas S_0 y S_1 son entradas de modo que determinan los modos de funcionamiento *desplazamiento a la derecha, desplazamiento a la izquierda* y *carga paralelo*, como indica la designación de dependencia 0/3 que sigue a la *M*. Esta designación representa los estados binarios de 0, 1, 2 y 3 en las entradas S_0 y S_1. Se establece una dependencia cuando uno de estos dígitos se utiliza como prefijo de otra entrada. El símbolo 1→/2← en la entrada de reloj significa lo siguiente: 1→ indica que se produce un desplazamiento a la derecha (de Q_0 a Q_3) cuando las entradas de modo S_0 y S_1 están en el estado binario 1 ($S_0 = 1$ y $S_1 = 0$); 2← indica que se produce un desplazamiento a la izquierda (de Q_3 a Q_0) cuando las entradas de modo están en el estado binario 2 ($S_0 = 0$ y $S_1 = 1$).

FIGURA 10.42
Símbolo lógico del 74HC164.

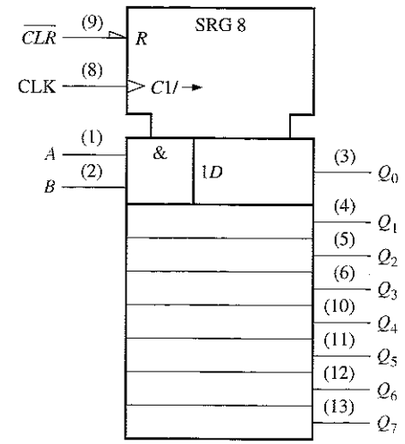
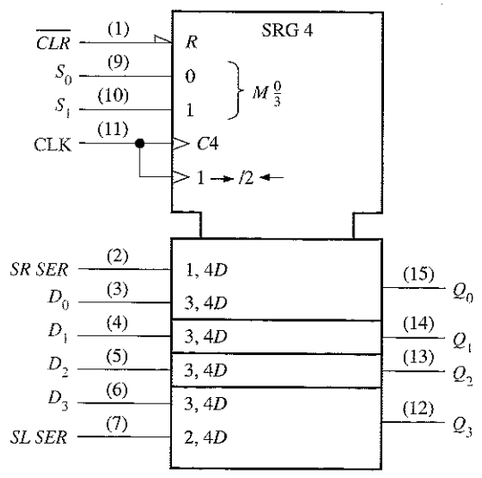


FIGURA 10.43
Símbolo lógico del 74HC194.



La entrada serie para el desplazamiento a la derecha (*SR SER)*, como indica 1, 4*D*, es dependiente del modo y dependiente del reloj. Las entradas paralelo (D_0, D_1, D_2 y D_3), como indica 3, 4*D*, son dependientes del modo (el prefijo 3 indica modo de carga paralelo) y dependientes del reloj. La entrada serie para desplazamiento a la izquierda (*SL SER)* también es dependiente del modo y de la señal de reloj, como indica 2, 4*D*.

Los cuatro modos del 74HC194 se resumen de la siguiente forma:

Mantiene el estado anterior:	$S_0 = 0, S_1 = 0$	(modo 0)
Desplazamiento a la derecha:	$S_0 = 1, S_1 = 0$	(modo 1, como en 1, 4*D*)
Desplazamiento a la izquierda:	$S_0 = 0, S_1 = 1$	(modo 2, como en 2, 4*D*)
Carga paralelo:	$S_0 = 1, S_1 = 1$	(modo 3, como en 3, 4*D*)

REVISIÓN DE LA SECCIÓN 10.10

1. En la Figura 10.43, ¿cuántas entradas son dependientes de las entradas de modo, si se está en el estado 0?
2. ¿Es la carga paralelo síncrona con la señal de reloj?

10.11 ■ APLICACIÓN A LOS SISTEMAS DIGITALES

En esta aplicación a los sistemas digitales, se va a trabajar con un sistema de seguridad para acceso. Este sistema permite desactivar las alarmas para poder entrar en un edificio. La desactivación del sistema se consigue introduciendo una secuencia específica de cuatro dígitos a través de un teclado numérico. El sistema se activa de nuevo pulsando un interruptor cuando se abandona el edificio. El sistema utiliza tres registros de desplazamiento y otros dispositivos que se han visto en los capítulos anteriores. También se emplea una memoria, aunque no se entra en detalles hasta que se estudien en el Capítulo 12.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir el funcionamiento global del sistema. ■ Explicar el propósito de los registros de desplazamiento y del resto de los dispositivos. ■ Determinar los dispositivos específicos que se usan en el sistema. ■ Desarrollar un diagrama lógico detallado a partir del diagrama lógico general. ■ Desarrollar un procedimiento de pruebas y de localización de averías del sistema.

Funcionamiento básico del sistema de seguridad para acceso

En la Figura 10.44 se presenta un esquema lógico preliminar del sistema. El sistema está formado por dos tarjetas de circuito impreso, el panel de interruptores y el teclado numérico. Esta sección se va a centrar en la tarjeta dedicada al código de acceso, y los dispositivos que utiliza son ya familiares. La tarjeta de memoria y el funcionamiento completo del sistema se estudian en el apartado "Aplicaciones a los sistemas digitales" del Capítulo 12.

El sistema de acceso de seguridad controla la activación y desactivación de los sensores y alarmas de un determinado edificio. La activación de los sensores y alarmas se consigue presionando un único pulsador designado por ACTIVACIÓN y, mientras éste esté pulsado, introduciendo cualquier dígito a través del teclado. La desactivación se realiza introduciendo a través del teclado una secuencia predeterminada de cuatro dígitos.

Cuando el sistema está activado, los registros de desplazamiento A y B y el contador de memoria se borran, pasando al estado en el que todas las posiciones son cero,

mientras el registro de desplazamiento C se inicializa en el estado 11110000. Como resultado, en la salida serie del registro de desplazamiento C, se presenta un nivel BAJO, que activa los sensores del sistema de alarma y hace que se encienda la luz de ACTIVACIÓN.

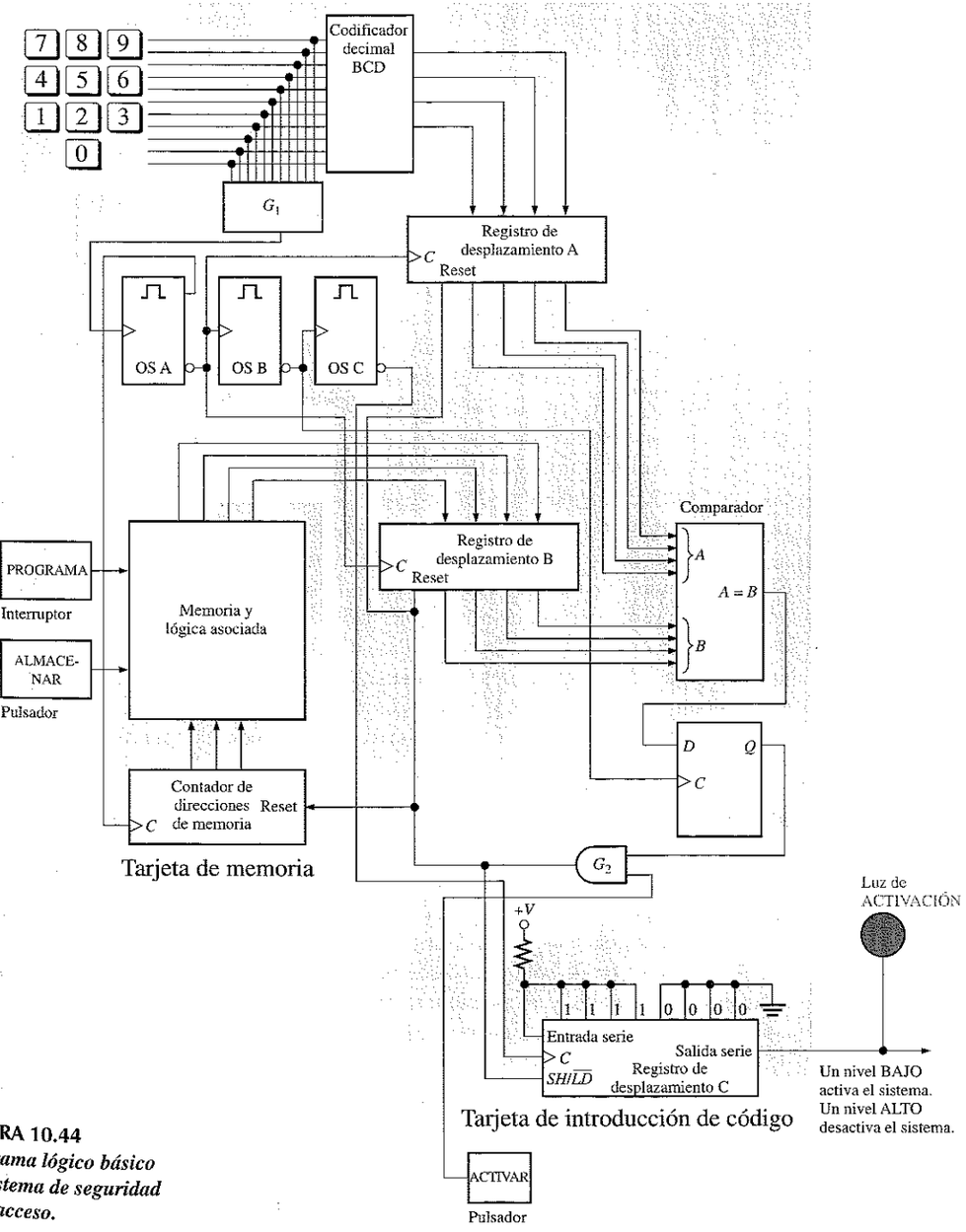


FIGURA 10.44
Diagrama lógico básico del sistema de seguridad para acceso.

Cuando se introduce a través del teclado el primer dígito del número de acceso, la correspondiente línea del teclado pasa a su nivel activo. El decodificador decimal-BCD genera el código BCD de 4 bits que corresponde al dígito decimal. Al mismo tiempo, el monoestable A se dispara y genera un impulso de reloj que hace que el código BCD de 4 bits del decodificador se cargue en el registro A y el primer dígito BCD de 4 bits de la memoria se cargue en el registro B. Estos códigos se presentan entonces en las entradas A y B del comparador. La memoria almacena los códigos BCD correspondientes a los números de acceso de 4 dígitos correctos.

Si el dígito correcto se ha introducido a través del teclado numérico, los cuatro bits de la entrada A del comparador serán iguales a los de la entrada B, por lo que la salida $A = B$ del comparador se pone a nivel ALTO. El impulso procedente del mononestable A también dispara al monoestable B. El impulso del monoestable B hace que se almacene el estado de la salida del comparador $A = B$ en el flip-flop, haciendo que su salida Q pase a nivel ALTO. El impulso procedente del monoestable C se aplica a la entrada de reloj del registro C y desplaza su contenido a la derecha, de manera que el registro contenga los bits 11111000. Dado que todavía hay un 0 en la última etapa (más a la derecha) del registro, el sistema de alarma permanece activado.

Después de introducir el segundo dígito correcto a través del teclado, el registro de desplazamiento C contiene el estado 11111100. Después de introducir el tercer dígito correcto, el registro de desplazamiento contendrá 11111110. Después de introducir el cuarto y último dígito correcto, el registro de desplazamiento contendrá todo unos. Cuando el 1 se ha desplazado hasta la última etapa, la salida serie del registro se pone a nivel ALTO y desactiva el sistema. La luz de ACTIVACIÓN se apaga en este momento y entonces se puede entrar en el edificio sin que salte la alarma.

Veamos ahora, muy brevemente, qué ocurre si se introduce un dígito incorrecto. Si uno cualquiera de los cuatro dígitos que se introducen es incorrecto, el código BCD generado no será igual al que está almacenado en la memoria. Por tanto, el comparador producirá un nivel BAJO en su salida $A = B$. Esto dará lugar a que se realice la carga paralelo en el registro de desplazamiento C de 11110000 y que los registros de desplazamiento A y B se borren y el contador de memoria se ponga todo a cero. A partir de este momento, debe comenzar de nuevo y reintroducir el número de 4 dígitos.

■ PRÁCTICAS DE SISTEMAS

El diagrama lógico básico de la Figura 10.44 indica la lógica general del sistema y no representa en detalle la lógica que se requiere para implementarlo. En las siguientes tres prácticas de sistemas, se analizará el funcionamiento básico de la lógica de la tarjeta de introducción de código de acceso, para asegurarse de que se ha entendido los requisitos funcionales y el funcionamiento básico del sistema. A continuación, se completará el diseño de la lógica seleccionando los dispositivos adecuados, y luego se desarrollará un esquema lógico detallado que incorpore los dispositivos seleccionados. Por último, se comparará su esquema lógico con la tarjeta del circuito implementada y se hará el desarrollo de un procedimiento de pruebas.

En la tarjeta de circuito impreso existe una serie de interconexiones entre taladros realizadas por la cara posterior de la tarjeta. Para simplificar, las conexiones a la alimentación y a tierra por la cara posterior de la tarjeta no se muestran.

■ PRÁCTICAS DE SISTEMAS 1: Análisis

■ ***Actividad 1. Codificador decimal-BCD***

(a) Establecer el propósito del codificador.
(b) Describir qué ocurre en las entradas del codificador cuando se introduce un dígito a través del teclado.
(c) Describir qué ocurre en las salidas del codificador cuando se introduce un dígito a través del teclado.

■ ***Actividad 2. Los circuitos de temporización***

(a) Establecer el propósito del bloque designado por G_1.
(b) Explicar cuándo y cómo se dispara el monoestable A.
(c) Establecer el objetivo de cada monoestable.

■ ***Actividad 3. Registros de desplazamiento A y B***

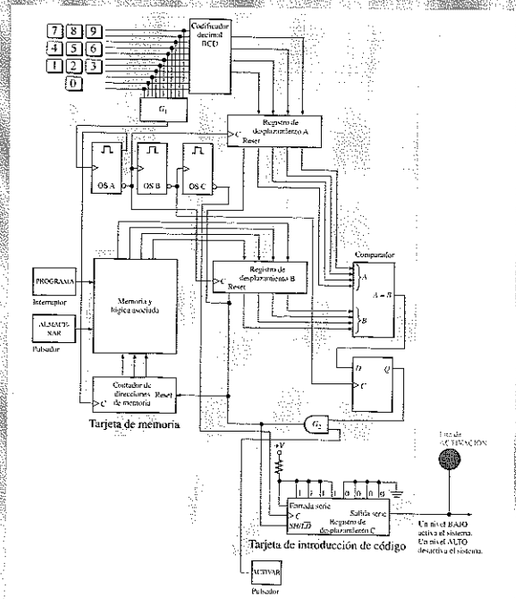
(a) Establecer el propósito de los registros de desplazamiento A y B.
(b) Determinar la capacidad de los registros de desplazamiento A y B (número de bits).
(c) Describir cuándo y cómo los datos paralelo se introducen en los registros de desplazamiento A y B.

■ ***Actividad 4. Comparador***

(a) Establecer el propósito del comparador.
(b) Definir cada uno de los conjuntos (A y B) de los cuatro bits de entrada.
(c) Definir qué ocurre en las salidas cuando las dos entradas son iguales.
(d) Describir qué ocurre en las salidas cuando las dos entradas son diferentes.

■ ***Actividad 5. Registro de desplazamiento C***

(a) Establecer el propósito del registro de desplazamiento C.
(b) Comentar la secuencia patrón de bits que se carga inicialmente en el registro y establecer su finalidad.
(c) Describir qué ocurre cuando la salida serie está a nivel BAJO. Describir qué ocurre cuando la salida serie está a nivel ALTO.
(d) Describir qué ocurre en el registro de desplazamiento C, cada vez que se introduce un dígito correcto a través del teclado.



■ PRÁCTICAS DE SISTEMAS 2: Diseño

Esta práctica se centra en el diseño de la tarjeta de introducción del código. Para obtener información sobre los dispositivos es necesario recurrir a los capítulos anteriores y/o a las hojas de características de los mismos.

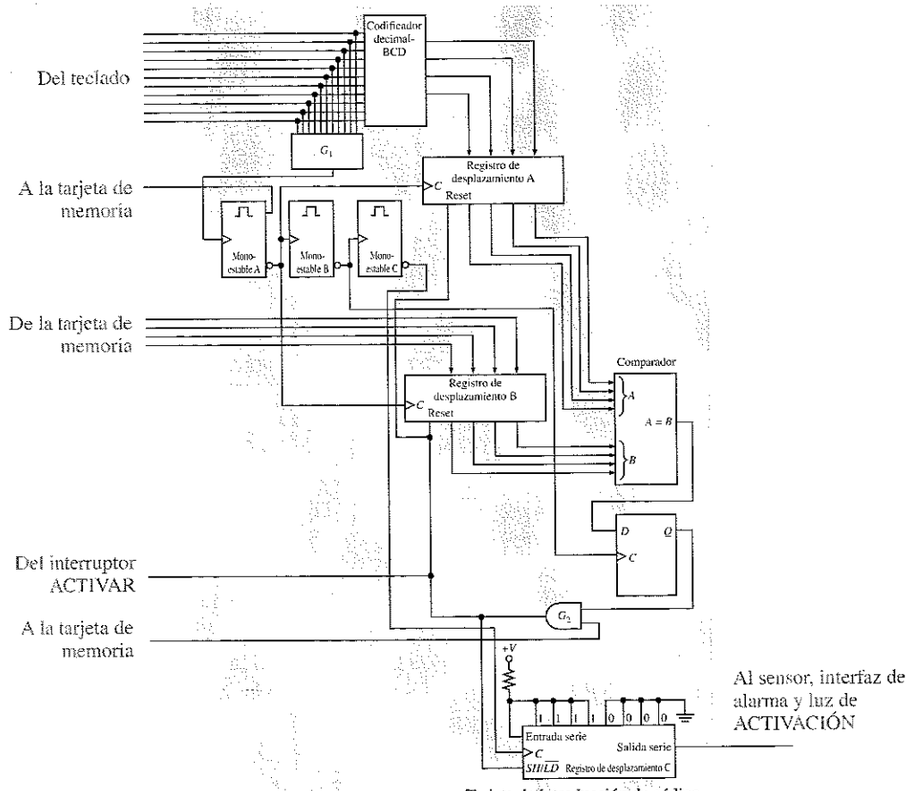
■ ***Actividad 1. Codificador decimal-BCD***

El teclado numérico que se utiliza en este sistema genera salidas activas a nivel BAJO. Cuando se pulsa una tecla, la línea correspondiente pasa a nivel BAJO y las demás líneas permanecen en estado ALTO. Especificar un CI decodificador que realice la interfaz adecuadamente con el teclado.

■ ***Actividad 2. Circuitos de temporización***

El diagrama lógico básico muestra el bloque G_1 para la puerta utilizada para disparar el monoestable A cuando se presiona una tecla. El monoestable A genera un impulso de 1 ms. El monoestable B genera un impulso de 1 ms, disparado por flanco posterior del impulso del monoestable A. El monoestable C produce un impulso de 1 ms, disparado por el flanco posterior del impulso del monoestable B.

(a) Especificar el tipo de puerta que debe utilizarse para implementar el bloque G_1.



Tarjeta de introducción de código

■ PRÁCTICAS DE SISTEMAS 2: Diseño (continuación)

(b) Especificar el tipo de CI monoestable y determinar los valores de los componentes externos requeridos, para que cada monoestable genere un impulso de 1 ms.

■ ***Actividad 3. Registros de desplazamiento A y B***

Especificar un CI registro de desplazamiento que pueda aceptar 4 bits en paralelo y proporcionar una salida paralelo. El registro de desplazamiento debe dispararse en el flanco posterior del impulso de reloj del monoestable A.

■ ***Actividad 4. Comparador***

Especificar un CI comparador que compare dos números de 4 bits y dé lugar a las tres salidas que se requieren.

■ ***Actividad 5. Registro de desplazamiento C***

Este registro debe almacenar la secuencia patrón de bits inicial 11110000, la cual se carga en paralelo. Debe disponer de una entrada serie y capacidad para realizar desplazamientos a la derecha, de modo que el 1 inicial se desplace a la derecha con cada impulso de reloj y el registro quede lleno de unos a partir de la entrada serie. La salida serie se utiliza para controlar el sistema de alarma y la luz de ACTIVACIÓN, lo cual requiere un circuito excitador externo a la tarjeta.

Especificar el dispositivo o dispositivos (circuitos integrados) para implementar este registro de desplazamiento.

■ ***Actividad 6. La lógica restante***

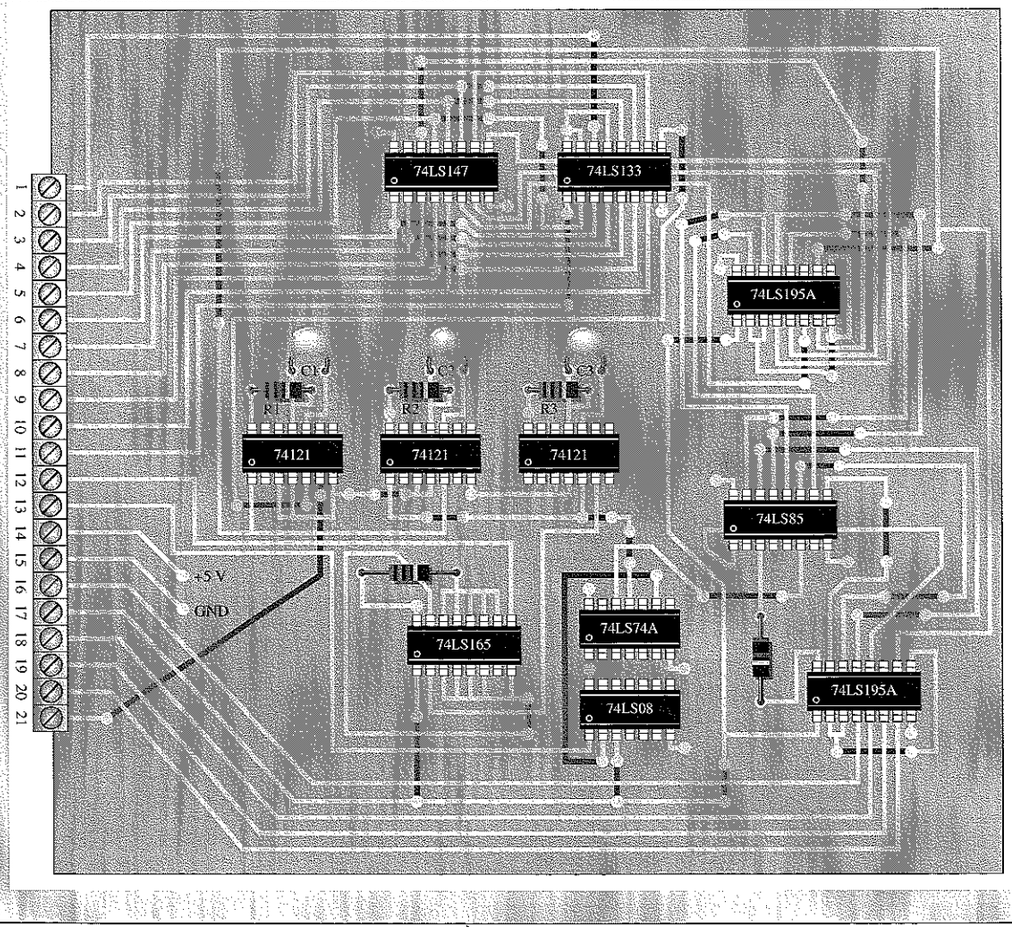
Especificar los tipos de circuitos integrados necesarios para la puerta AND y el flip-flop.

■ ***Actividad 7.*** Desarrollar un diagrama lógico detallado utilizando los dispositivos especificados en las actividades precedentes. Mostrar todos los detalles, incluyendo la numeración de los pines de los dispositivos.

■ PRÁCTICA DE SISTEMAS 3: Verificación y pruebas

- ***Actividad 1.*** Dibujar el diagrama lógico para la tarjeta de introducción del código mostrada más abajo y compararlo con el diagrama generado en la actividad 7 de la Práctica de sistemas 2. Comentar cualquier diferencia entre su diseño y el diseño de esta tarjeta, y verificar que ambos funcionan.
- ***Actividad 2.*** Identificar las salidas y entradas del conector de la tarjeta del circuito. Dispone de más entradas y salidas que las mostradas en el diagrama básico de la Práctica de sistemas 2, ya que existen otros requisitos adicionales para llevar a cabo la interfaz con la tarjeta de memoria, que se verá en el Capítulo 12.
- ***Actividad 3.*** Desarrollar un procedimiento de prueba para comprobar la tarjeta mostrada.

El circuito completo de la tarjeta para la lógica de introducción de código modificado se encuentra en el archivo SA10.EWB del CD-ROM EWB. Utilícese el EWB para simular una prueba de este circuito.



REVISIÓN DE LA SECCIÓN 10.11

1. ¿Cuáles son los pasos requeridos para activar el sistema?
2. ¿Cómo se desactiva el sistema?
3. ¿Qué ocurre si se introduce un dígito incorrecto después de dos dígitos correctos?

En este punto puede estudiar el Capítulo 15, "Tecnología de circuitos integrados", parcial o totalmente. El examen de este capítulo no es un prerrequisito para abordar cualquier otro tema tratado en este texto, y su omisión total o parcial no afectará al resto de los temas.

■ RESUMEN

- Los tipos básicos de movimiento de los datos en los registros de desplazamiento se ilustran en la Figura 10.45.
- Los contadores basados en registros de desplazamiento son registros de desplazamiento con realimentación, que disponen de secuencias especiales. Ejemplos de ellos son el contador Johnson y el contador en anillo.
- La secuencia del contador Johnson tiene $2n$ estados, donde n es el número de etapas.
- La secuencia del contador en anillo tiene n estados.

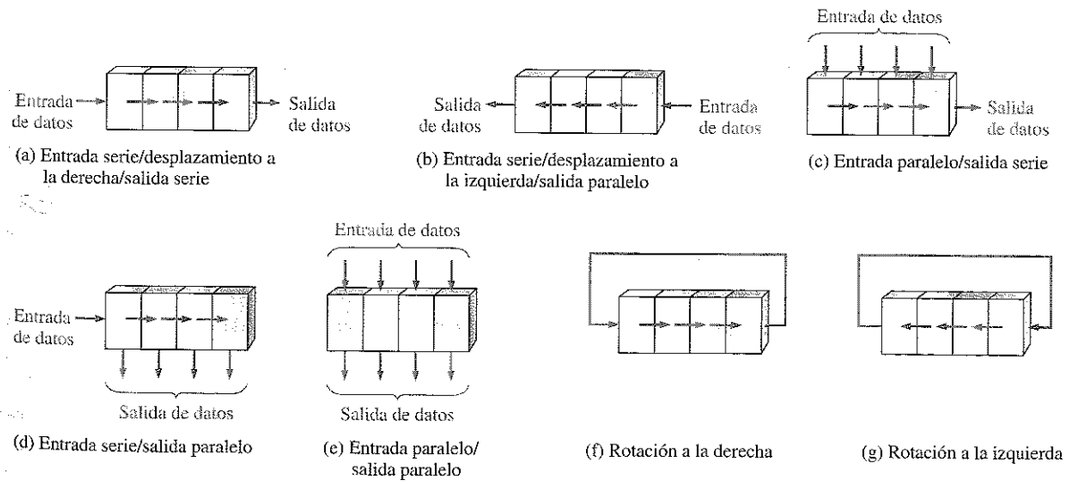


FIGURA 10.45

■ PALABRAS CLAVE

Estos términos también se encuentran en el glosario del final del libro.

Bidireccional Que posee dos direcciones. En un registro de desplazamiento bidireccional, los datos almacenados se pueden desplazar a la derecha o a la izquierda.

Carga Introducir datos en un registro de desplazamiento.

Contador en anillo Registro en el que un determinado patrón de unos y ceros circula continuamente.

Contador Johnson Un tipo de registro en el que se desplaza un patrón específico de unos y ceros a través de las etapas, creando una secuencia patrón única de bits.

Desplazar Mover datos binarios de una etapa a otra dentro de un registro de desplazamiento o de otro dispositivo de almacenamiento; mover datos binarios dentro o fuera de un dispositivo.

Etapa Elemento de almacenamiento en un registro.

Registro de desplazamiento universal Registro que tiene capacidad de entradas y salidas serie y paralelo.

■ AUTOTEST

1. Una etapa de un registro de desplazamiento está formada por:
   (a) un latch (b) un flip-flop
   (c) un byte de almacenamiento (d) cuatro bits de almacenamiento
2. Para desplazar en serie un byte de datos en un registro de desplazamiento, es necesario:
   (a) un impulso de reloj (b) un impulso de carga
   (c) ocho impulsos de reloj (d) un impulso de reloj para cada 1 que contiene el dato
3. Para cargar en paralelo un byte de datos en un registro de desplazamiento con una carga síncrona, es necesario:
   (a) un impulso de reloj (b) un impulso de reloj para cada 1 que contiene el dato
   (c) ocho impulsos de reloj (d) un impulso de reloj para cada 0 que contiene el dato
4. El grupo de bits 10110101 se desplaza en serie (primer bit más a la derecha) a la salida paralelo de 8 bits de un registro de desplazamiento, el cual tiene el estado inicial 11100100. Después de dos impulsos de reloj, el contenido del registro es:
   (a) 01011110 (b) 10110101 (c) 01111001 (d) 00101101
5. Con una frecuencia de reloj de 100 kHz, ocho bits se pueden introducir en serie en un registro de desplazamiento en:
   (a) 80 μs (b) 8 μs (c) 80 ms (d) 10 μs
6. Con una frecuencia de reloj de 1 MHz, ocho bits se pueden introducir en paralelo en un registro de desplazamiento en:
   (a) 8 μs (b) en un tiempo igual al retardo de propagación de ocho flip-flops
   (c) 1 μs (d) en un tiempo igual al retardo de propagación de un flip-flop
7. Un contador Johnson de módulo 10 requiere
   (a) diez flip-flops (b) cuatro flip-flops
   (c) cinco flip-flops (d) doce flip-flops
8. Un contador en anillo de módulo 10 requiere como mínimo
   (a) diez flip-flops (b) cinco flip-flops
   (c) cuatro flip-flops (d) doce flip-flops
9. Cuando se utiliza un registro de desplazamiento de 8 bits con entrada y salida serie, para obtener un retardo de 24 μs, la frecuencia de reloj debe ser
   (a) 41,67 kHz (b) 333 kHz (c) 125 kHz (d) 8 MHz

10. El propósito del contador en anillo del circuito codificador de teclado de la Figura 10.38 es:
    (a) aplicar secuencialmente un nivel ALTO a cada fila para detectar la pulsación de una tecla
    (b) proporcionar los impulsos de disparo del registro de código de tecla
    (c) aplicar secuencialmente un nivel BAJO a cada fila para detectar la pulsación de una tecla
    (d) invertir secuencialmente la polarización de los diodos de cada fila.

■ PROBLEMAS

SECCIÓN 10.1 Funciones básicas de los registros de desplazamiento

1. ¿Por qué se consideran los registros de desplazamiento dispositivos básicos de memoria?
2. ¿Cuál es la capacidad de almacenamiento de un registro que puede contener dos bytes de datos?

SECCIÓN 10.2 Registros de desplazamiento con entrada y salida serie

3. Para las señales de entrada de datos y de reloj de la Figura 10.46, determinar los estados de cada flip-flop del registro de desplazamiento de la Figura 10.3 y dibujar las formas de onda de salida. Suponer que, inicialmente, el registro contiene todo unos.
4. Resolver el problema 3 para las formas de onda de la Figura 10.47.

FIGURA 10.46

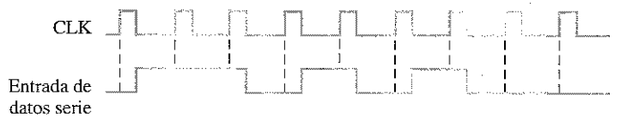


FIGURA 10.47

5. ¿Cuál es el estado del registro de la Figura 10.48 después de cada impulso de reloj, si el estado inicial es 101001111000?
6. Para el registro de desplazamiento con entrada serie y salida serie, determinar la forma de onda de la salida de datos para la señal de reloj y la entrada de datos de la Figura 10.49. Suponer que, inicialmente, se borra el contenido del registro.

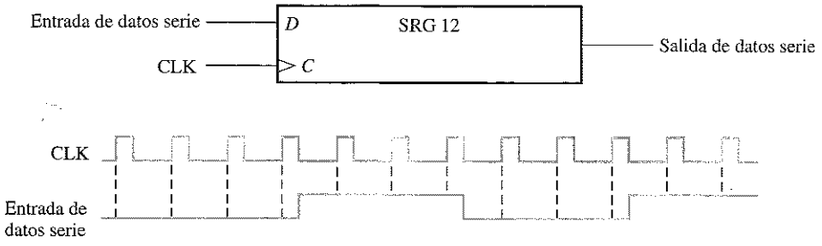


FIGURA 10.48

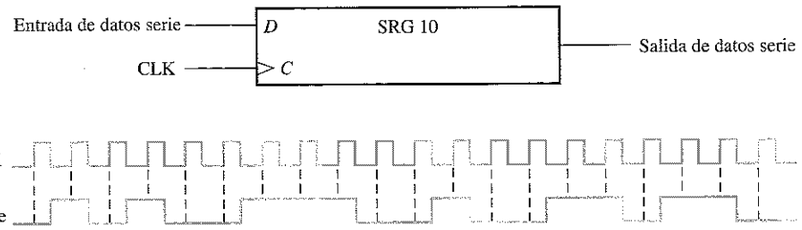
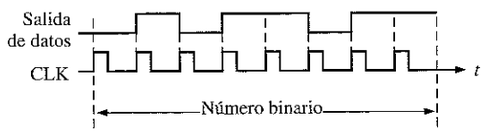


FIGURA 10.49

7. Resolver el problema 6 para las formas de onda de la Figura 10.50.
8. Un registro de desplazamiento de entrada serie-salida paralelo disparado por flanco anterior tiene la forma de onda de la salida de datos mostrada en la Figura 10.51. ¿Qué número binario se almacena en el registro de 8 bits, si el primer bit de datos que sale (el que está más a la izquierda) es el LSB?

FIGURA 10.50

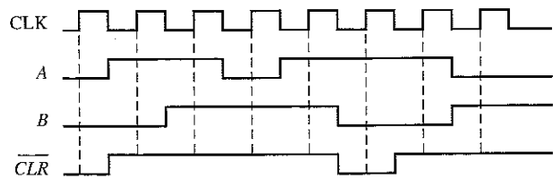
FIGURA 10.51



SECCIÓN 10.3 Registros de desplazamiento con entrada serie-salida paralelo

9. Dibujar un diagrama de tiempos completo que muestre las salidas paralelo del registro de desplazamiento de la Figura 10.8. Utilizar las formas de onda de la Figura 10.49, estando inicialmente borrado el registro.
10. Resolver el problema 9 para las formas de onda de la Figura 10.50.
11. Desarrollar las salidas Q_0 a Q_7 para un registro de desplazamiento 74HC164, siendo la señal de entrada la mostrada en la Figura 10.52.

FIGURA 10.52



SECCIÓN 10.4 Registros de desplazamiento con entrada paralelo-salida serie

12. Al registro de desplazamiento de la Figura 10.53(a) se le aplican las señales de entrada *SHIFT/LOAD* y CLK mostradas en la parte (b). La entrada de datos serie (*SER*) está a 0. Las entradas de datos paralelo son $D_0 = 1$, $D_1 = 0$, $D_2 = 1$ y $D_3 = 0$. Dibujar la forma de onda de la salida de datos en función de las entradas.
13. Las formas de onda de la Figura 10.54 se aplican al registro de desplazamiento 74HC165. Todas las entradas paralelo están a 0. Determinar la forma de onda de la salida Q_7.

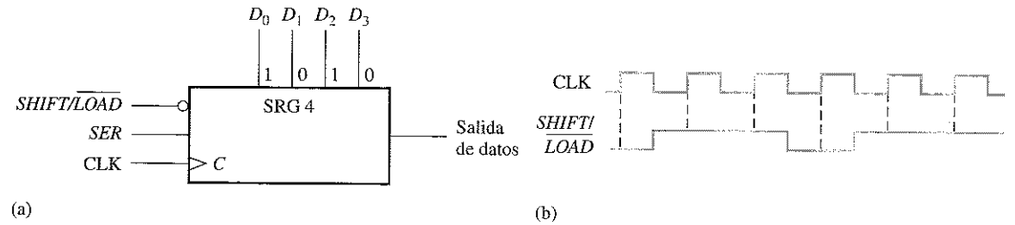
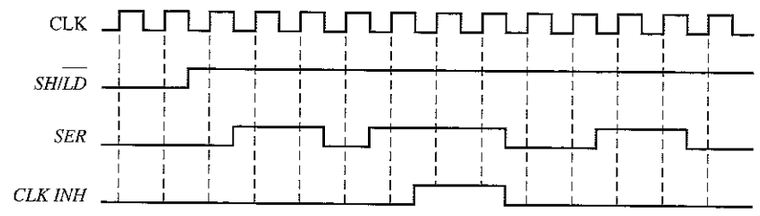


FIGURA 10.53

FIGURA 10.54



14. Resolver el problema 13 si las entradas paralelo están todas a 1.
15. Resolver el problema 13 si se invierte la entrada *SER*.

SECCIÓN 10.5 Registros de desplazamiento con entrada y salida paralelo

16. Determinar todas las formas de onda de salida Q para el registro de desplazamiento de 4 bits 74HC195, cuando las entradas son las indicadas en la Figura 10.55.

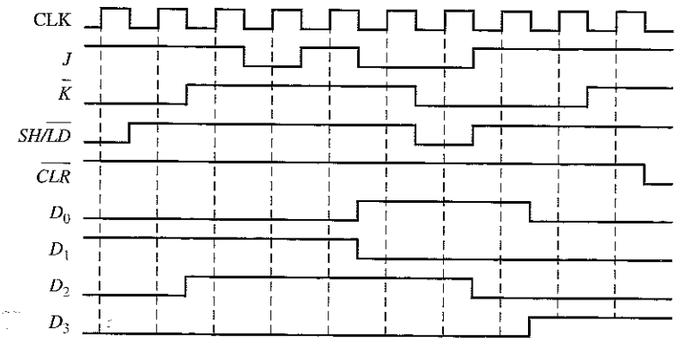


FIGURA 10.55

17. Resolver el problema 16 si se invierte la entrada *SH/LD* e, inicialmente, se borra el contenido del registro.

18. Utilizar dos registros de desplazamiento 74HC195 para formar un registro de desplazamiento de 8 bits. Mostrar las conexiones que se requieren.

SECCIÓN 10.6 Registros de desplazamiento bidireccionales

19. Para el registro bidireccional de 8 bits de la Figura 10.56, determinar el estado del registro después de cada impulso de reloj para la señal de control *RIGHT/LEFT* dada. Un nivel ALTO en esta entrada activa un desplazamiento a la derecha, y un nivel BAJO lo activa hacia la izquierda. Suponer que, inicialmente, el registro almacena en binario el número decimal setenta y seis, estando en la posición más a la derecha el LSB. La línea de entrada de datos está a nivel BAJO.

20. Resolver el problema 19 para las señales de la Figura 10.57.

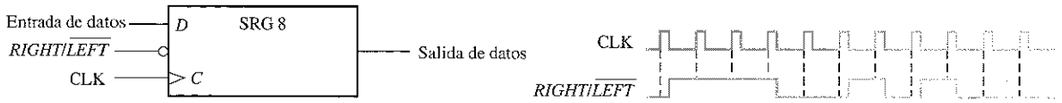


FIGURA 10.56

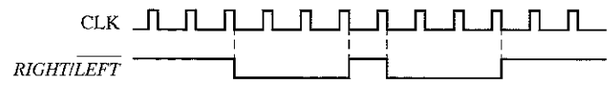


FIGURA 10.57

21. Utilizar dos registros de desplazamiento bidireccionales de 4 bits 74HC194, para crear un registro de desplazamiento bidireccional de 8 bits. Indicar las conexiones.

22. Determinar las salidas Q de un 74HC194 a partir de las entradas mostradas en la Figura 10.58. Las entradas D_0, D_1, D_2 y D_3 están a nivel ALTO.

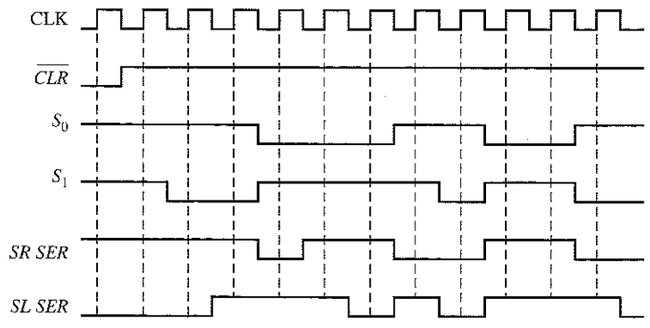


FIGURA 10.58

SECCIÓN 10.7 Contadores basados en registros de desplazamiento

23. ¿Cuántos flip-flops se requieren para implementar cada una de las siguientes configuraciones de un contador Johnson?

(a) módulo 6 **(b)** módulo 10 **(c)** módulo 14 **(d)** módulo 16

24. Dibujar el diagrama lógico para un contador Johnson de módulo 18. Realizar el diagrama de tiempos y escribir la secuencia en forma de tabla.

25. Para el contador en anillo de la Figura 10.59, dibujar la señal de salida de cada flip-flop en relación con la señal de reloj. Suponer que, inicialmente, FF0 está en estado SET y los demás en estado RESET. Considerar al menos diez impulsos de reloj.

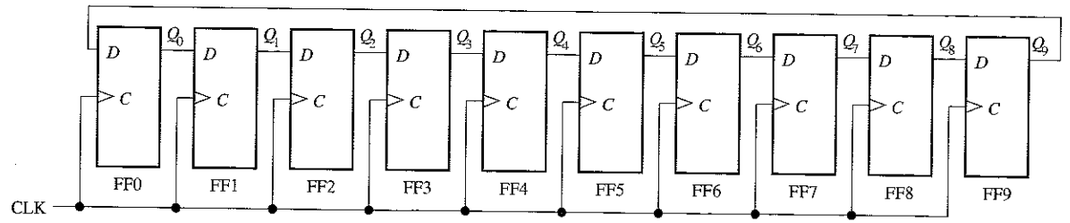


FIGURA 10.59

26. A partir de la secuencia patrón mostrada en la Figura 10.60, determinar el contador en anillo e indicar cómo se puede inicializar para generar la señal indicada en la salida Q_9. En el impulso de reloj 16 (CLK16) la secuencia patrón se repite.

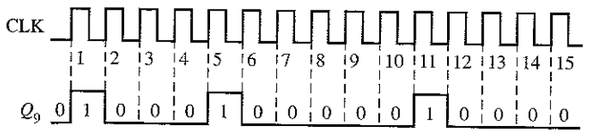


FIGURA 10.60

SECCIÓN 10.8 Aplicaciones de los registros de desplazamiento

27. Utilizar registros de desplazamiento de 4 bits 74HC195 para implementar un contador en anillo de 16 bits. Indicar las conexiones.

28. ¿Cuál es el propósito de la entrada de alimentación $\overline{LOAD}$ de la Figura 10.38?

29. En el esquema de la Figura 10.38 ¿qué ocurre cuando se presionan simultáneamente dos teclas?

SECCIÓN 10.9 Localización de averías

30. A partir de las formas de onda de la Figura 10.61(a), determinar el problema que más probablemente se producirá en el registro mostrado en la parte (b) de la figura.

31. El registro de desplazamiento con entrada paralelo y salida serie de la Figura 10.12 está en el estado en que $Q_0\,Q_1\,Q_2\,Q_3 = 1001$ y en la entrada se carga $D_0\,D_1\,D_2\,D_3 = 1010$. Cuando la entrada *SHIFT/* $\overline{LOAD}$ está a nivel ALTO, los datos que se muestran en la Figura 10.62 aparecen secuencialmente en la salida. ¿Es correcto este funcionamiento? Si no lo es, ¿cuál es el fallo más probable?

32. Como hemos visto, en el registro bidireccional de la Figura 10.19, los datos se desplazan hacia la derecha pero no hacia la izquierda. ¿Cuál es el fallo más probable?

33. Para el codificador de teclado de la Figura 10.38, enumerar los posibles fallos para cada uno de los siguientes síntomas:

(a) El estado del registro del código de tecla no cambia cuando se pulsa cualquier tecla.
(b) El estado del registro del código de tecla no cambia cuando se pulsa cualquier tecla de la tercera fila. Cuando se pulsa cualquiera de las restantes teclas se genera un código correcto.
(c) El estado del registro del código de tecla no cambia cuando se pulsa cualquier tecla de la primera columna. Cuando se pulsa cualquiera de las restantes teclas se genera un código correcto.
(d) Cuando se pulsa cualquier tecla de la segunda columna, los tres bits de la izquierda del código de tecla (Q_0 Q_1 Q_2) son correctos pero los tres bits de la derecha son todos unos.

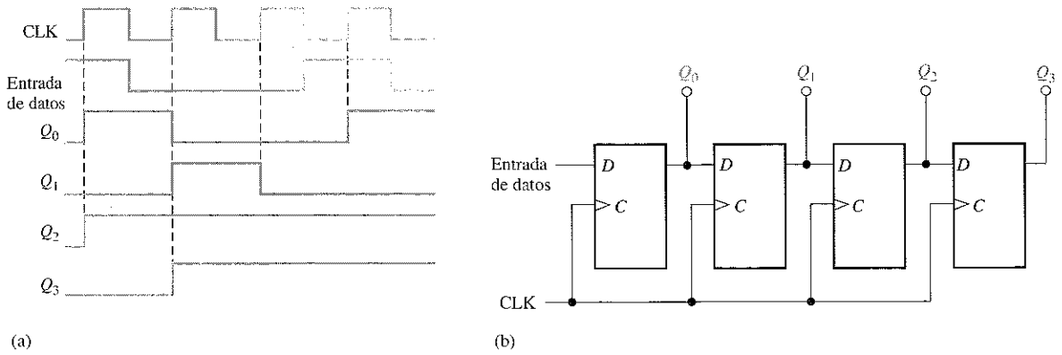
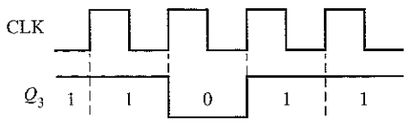


FIGURA 10.61

FIGURA 10.62



34. Desarrollar un procedimiento de prueba para probar el codificador de teclado de la Figura 10.38. Especificar el procedimiento paso a paso, indicando el código de salida del registro de código de tecla que debería observarse en cada paso de las pruebas.

35. ¿Qué síntomas se observan si se producen los siguientes fallos en el convertidor serie-paralelo de la Figura 10.33?

(a) La salida de la puerta AND se mantiene en estado ALTO.
(b) La salida del generador de reloj se mantiene en estado BAJO.
(c) La tercera etapa del registro de entrada de datos se mantiene en estado SET.
(d) La salida de fin de cuenta del contador se mantiene en estado ALTO.

SECCIÓN 10.11 Aplicación a los sistemas digitales

36. Dibujar el diagrama de tiempos de las tres salidas de los monoestables, como se especifica en la Práctica de sistemas 2.

37. Suponer que el código de acceso es 1939. Determinar los estados de los registros de desplazamiento A y C después de introducir el segundo dígito correcto.

38. Suponer que el código de acceso es 7646 y que se introduce el código 7645. Determinar los estados de los registros de desplazamiento A y C después de introducir cada uno de los dígitos.

39. En el sistema de seguridad para acceso se ha producido un problema tal, que el sistema no se puede desactivar (la salida serie del registro de desplazamiento C permanece en estado BAJO). Después de comprobar varios puntos, se encuentra que el monoestable A nunca se dispara. ¿Cuáles son los posibles fallos?

40. Al igual que en el problema 39, el sistema de seguridad para acceso no se puede desactivar. Sin embargo, esta vez todos los monoestables se disparan correctamente. ¿Qué dispositivos podrían ser, en este caso, los causantes del fallo?

Problemas especiales de diseño

41. Especificar los dispositivos que se pueden utilizar para implementar el convertidor de datos serie-paralelo de la Figura 10.33. Dibujar el diagrama lógico completo mostrando cualquier modificación necesaria para acomodarse a los dispositivos específicos utilizados.

42. Modificar el convertidor serie-paralelo de la Figura 10.33, para conseguir una conversión de 16 bits.

43. Diseñar un convertidor de datos paralelo-serie de 8 bits que produzca el formato de datos de la Figura 10.34. Realizar el diagrama lógico y especificar los dispositivos.

44. Diseñar un circuito de activación de $\overline{LOAD}$ para el codificador de teclado de la Figura 10.38. Este circuito debe generar impulsos de corta duración a nivel BAJO cuando se activa el interruptor de alimentación.

45. Implementar el generador de secuencias patrón de pruebas utilizado en la Figura 10.40 para localizar las averías en el convertidor serie-paralelo.

46. Revisar el sistema de control y de recuento de pastillas introducido en el Capítulo 1, y con el que se ha trabajado en las secciones dedicadas a las aplicaciones de sistemas de los Capítulos 2, 3 y 4. Utilizando los conocimientos adquiridos en este capítulo. implementar los registros A y B en este sistema utilizando circuitos integrados específicos.

Localización de fallos con el EWB

47. Abrir el archivo PR10_47.EWB y probar el registro de desplazamiento de 4 bits con el fin de determinar si existe un fallo. Si es posible, identificarlo.

48. Abrir el archivo PR10_48.EWB y probar el registro de desplazamiento de 8 bits de entrada serie-salida paralelo 74164, con el fin de determinar si existe un fallo. Si existe, identificarlo si es posible.

49. Abrir el archivo PR10_49.EWB y probar el registro de desplazamiento de 8 bits de carga paralelo 74165, con el fin de determinar si existe un fallo. Si existe, identificarlo si es posible.

50. Abrir el archivo PR10_50.EWB y probar el registro de desplazamiento de 4 bits de acceso paralelo 74195, con el fin de determinar si existe un fallo. Si existe, identificarlo si es posible.

51. Abrir el archivo PR10_51.EWB y probar el contador en anillo de 10 bits, con el fin de determinar si existe un fallo. Si existe, identificarlo si es posible.

■ RESPUESTAS A LAS REVISIONES DE CADA SECCIÓN

SECCIÓN 10.1

1. Un contador tiene una secuencia específica de estados, pero un registro de desplazamiento no.
2. El almacenamiento y el movimiento de datos son dos funciones de un registro de desplazamiento.

SECCIÓN 10.2

1. FF0: entrada de datos a J_0, $\overline{\text{entrada de datos}}$ a K_0; FF1: Q_0 a J_1, $\overline{Q}_0$ a K_1; FF2: Q_1 a J_2, $\overline{Q}_1$ a K_2; FF3: Q_2 a J_3, $\overline{Q}_2$ a K_3
2. Ocho impulsos de reloj.

SECCIÓN 10.3

1. 0100 después de 2 impulsos de reloj.
2. Se toma la salida serie del flip-flop más a la derecha para operación de salida serie.

SECCIÓN 10.4

1. Cuando *SHIFT/$\overline{LOAD}$* está a nivel ALTO, los datos se desplazan a la derecha, un bit por impulso de reloj. Cuando*SHIFT/$\overline{LOAD}$* está a nivel BAJO, los datos en las entradas paralelo se cargan en el registro.
2. La operación de carga paralelo es asíncrona, por lo que no depende de la señal de reloj.

SECCIÓN 10.5

1. Las salidas de datos son 1001
2. $Q_0 = 1$ después de un impulso de reloj

SECCIÓN 10.6

1. 11111 después de cinco impulsos de reloj.

SECCIÓN 10.7

1. La secuencia del contador Johnson de 8 bits tiene dieciséis estados.
2. Para un contador Johnson de 3 bits: 000, 100, 110, 111, 011, 001, 000

SECCIÓN 10.8

1. 625 exploraciones/segundo
2. $Q_5\,Q_4\,Q_3\,Q_2\,Q_1\,Q_0 = 011011$
3. Los diodos proporcionan caminos unidireccionales para poner las filas a nivel BAJO, y evitar que los niveles ALTOS en las líneas de FILA se conecten a la matriz de interruptores. Las resistencias conectan las líneas de COLUMNA a nivel ALTO.

SECCIÓN 10.9

1. Se usa una entrada de prueba para que el circuito pase por todos sus estados.
2. Comprobar la entrada de esta parte del circuito. Si la señal en esta entrada es correcta, el fallo queda aislado en la circuitería entre la entrada correcta y la salida incorrecta.

SECCIÓN 10.10

1. Ninguna entrada depende de las entradas de modo que estén en estado 0.
2. Sí, el terminal paralelo es síncrono con el reloj, como indica la etiqueta 4*D*.

SECCIÓN 10.11

1. Presionar el interruptor ACTIVAR, después pulsar cualquier tecla para activar el sistema.
2. Introducir el número de 4 dígitos correcto para desactivar el sistema.
3. El sistema se reinicializa.

■ RESPUESTAS A LOS PROBLEMAS RELACIONADOS DE LOS EJEMPLOS

10.1. Véase la Figura 10.63
10.2. El estado del registro después de tres impulsos adicionales de reloj es 0000
10.3. Véase la Figura 10.64
10.4. Véase la Figura 10.65
10.5. Véase la Figura 10.66
10.6. $f = 1/3\ \mu s = 333$ kHz

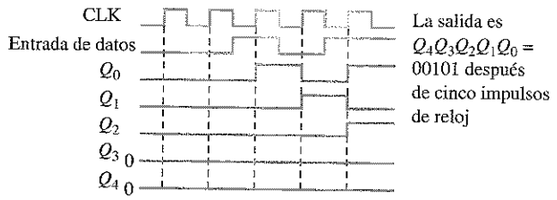


FIGURA 10.63

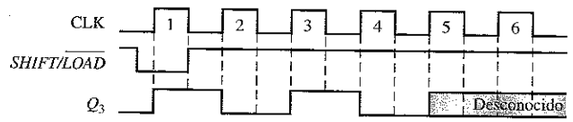


FIGURA 10.64

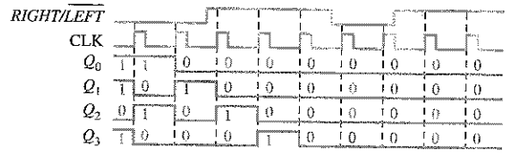


FIGURA 10.65

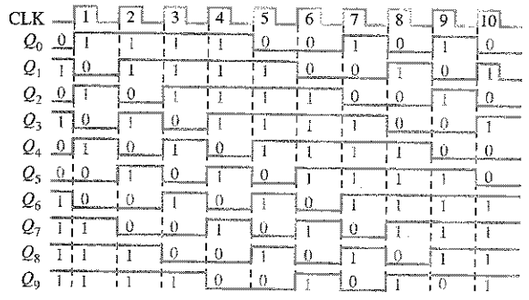


FIGURA 10.66

■ RESPUESTAS AL AUTOTEST

1. (b) **2.** (c) **3.** (a) **4.** (c) **5.** (a) **6.** (d) **7.** (c) **8.** (a)
9. (b) **10.** (c)

11

APLICACIONES DE LÓGICA SECUENCIAL CON DISPOSITIVOS LÓGICOS PROGRAMABLES

11.1 La celda OLMC al completo
11.2 Selección del modo de las OLMC
11.3 Implementación de registros de desplazamiento con PLD
11.4 Implementación de contadores con PLD
11.5 Implementación de sistemas mediante PLD
11.6 Aplicación a los sistemas digitales

Simulación por computadora de circuitos digitales:
El tutorial de Electronics Workbench (EWB) se encuentra en la dirección http://www.prenhall.com/floyd.

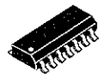
■ OBJETIVOS DEL CAPÍTULO

- ❑ Describir las OLMC en la GAL22V10 y la GAL16V8.
- ❑ Explicar la diferencia entre el modo secuencial y el modo combinacional.
- ❑ Desarrollar archivos de entrada en ABEL para el diseño de registros de desplazamiento.
- ❑ Utilizar el lenguaje ABEL para implementar contadores en un PLD.
- ❑ Utilizar diagramas de estado en ABEL para implementar sistemas lógicos en dispositivos PLD.
- ❑ Implementar la lógica secuencial del sistema de control de semáforos utilizando una GAL.

■ INTRODUCCIÓN

En el Capítulo 7 se han cubierto los fundamentos de los dispositivos lógicos programables, haciendo especial hincapié en la utilización de los PLD para la implementación de lógica combinacional. El estudio de los PLD se continúa en este capítulo con una introducción al modo secuencial y a las aplicaciones de la lógica secuencial. Si le es necesario, repase el Capítulo 7 durante o antes de comenzar con el estudio de este capítulo. Al igual que el Capítulo 7, el estudio de este capítulo puede omitirse sin que afecte al resto de los temas.

Como en el Capítulo 7, se utiliza el lenguaje de programación ABEL para ilustrar la implementación de la lógica secuencial, aunque el método puede aplicarse fácilmente a CUPL o cualquier otro lenguaje similar para circuitos PLD.

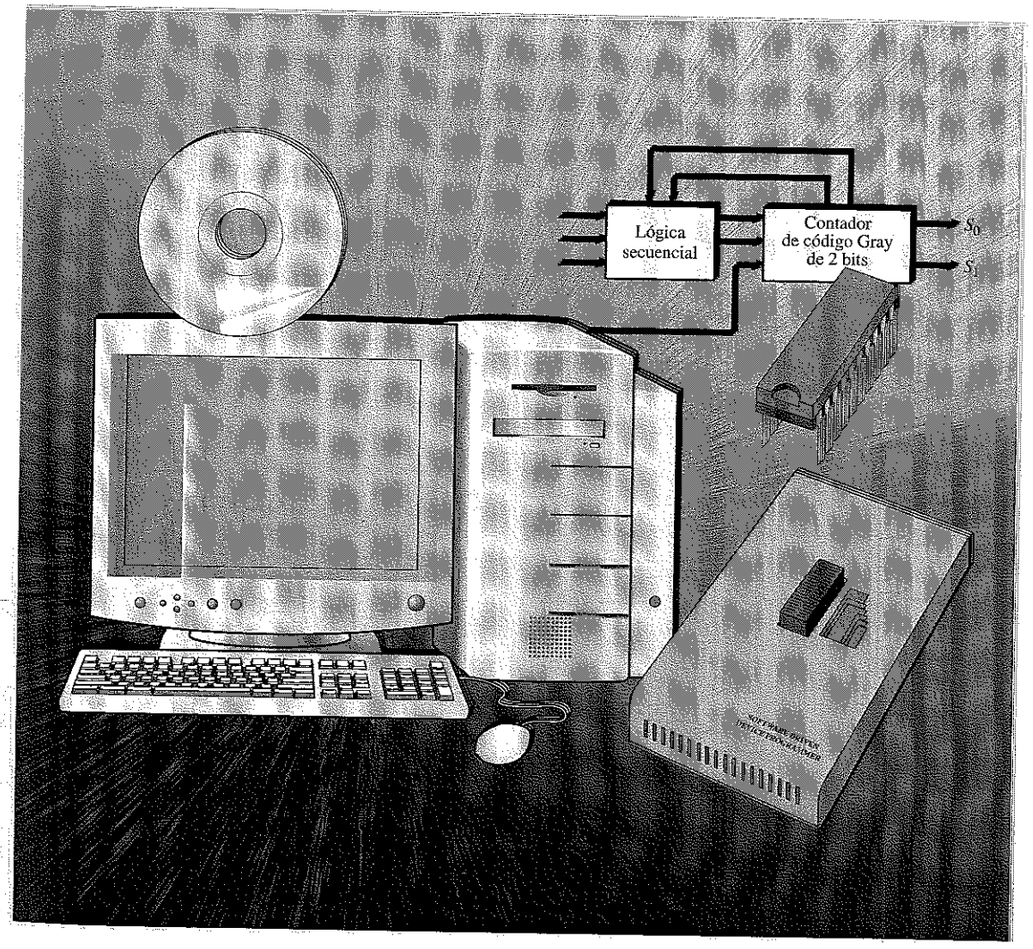


DISPOSITIVOS ESPECÍFICOS

GAL16V8 GAL22V10

■ APLICACIÓN A LOS SISTEMAS DIGITALES

Esta aplicación ilustra los conceptos que se van a aprender en el capítulo. La etapa de la lógica secuencial del sistema de control de semáforos del Capítulo 9 se va a rediseñar, utilizando un dispositivo lógico programable, y se implementa el sistema completo, a excepción de los circuitos de temporización, empleando una única GAL22V10.



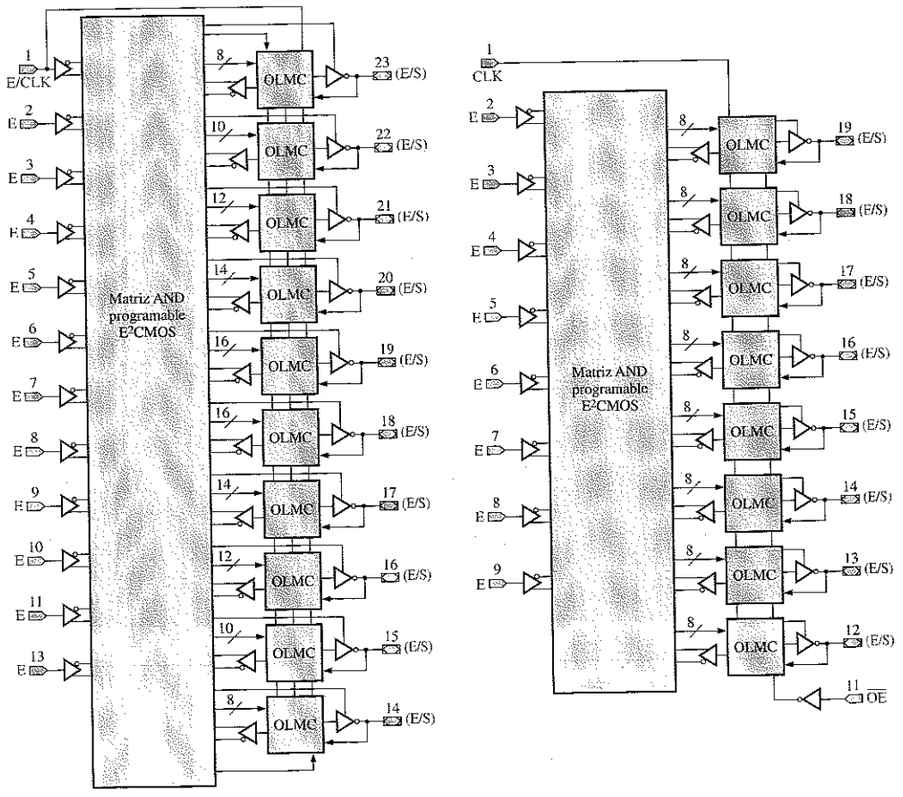
11.1 ■ LA CELDA OLMC AL COMPLETO

En el Capítulo 7 se examinaron las macroceldas lógicas de salida (OLMC, output logic macrocells) de la GAL22V10 y de la GAL16V8, en términos de sus modos combinacionales, y los flip-flops que contenían sólo se mencionaron brevemente. En esta sección, estudiaremos la OLMC completa incluyendo su flip-flop, utilizando de nuevo la GAL22V10 y la GAL16V8 como dispositivos PLD representativos.

Al finalizar el capítulo, el lector deberá ser capaz de:

- Conocer el funcionamiento básico de una OLMC
- Explicar qué son las celdas globales y locales

Los diagramas de bloques de la GAL22V10 y de la GAL16V8 se han introducido en el Capítulo 7, y se repiten aquí en la Figura 11.1. La GAL22V10 tiene diez macroceldas lógicas de salida (OLMC) y la GAL16V8 tiene ocho.



(a) GAL22V10

(b) GAL16V8

FIGURA 11.1
Diagramas de bloques de las GAL.

La macrocelda lógica de salida (OLMC) de la GAL22V10

Como se ha dicho en el Capítulo 7, una OLMC contiene circuitos lógicos programables, que se pueden configurar como entrada o salida **combinacional**, o como salida **secuencial**. En el modo secuencial, los pines que se definen como salidas secuenciales proceden de la salida Q de un flip-flop. Como ya sabemos, las OLMC se configuran internamente de forma automática, mediante la programación de un conjunto de celdas que están separadas de las celdas de la matriz lógica.

Diagrama lógico. La Figura 11.2 muestra el diagrama lógico básico de la OLMC de la GAL22V10. Las entradas a la puerta OR de la OLMC procedentes de las puertas AND conectadas a la matriz varían de diez a dieciséis. El resto de la lógica consiste en un flip-flop y dos multiplexores.

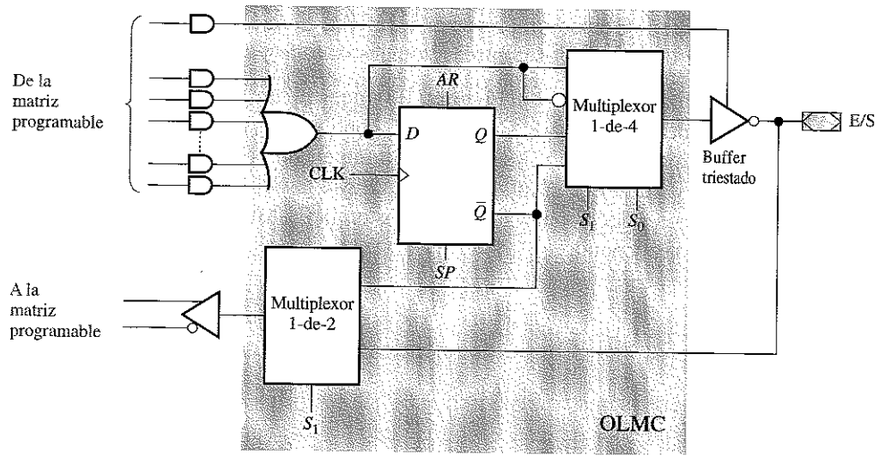


FIGURA 11.2
OLMC de la GAL22V10.

El multiplexor 1-de-4 conecta una de sus cuatro líneas de entrada al buffer de tres estados de salida en función del estado de las dos entradas de selección, S_0 y S_1. Las entradas del multiplexor 1-de-4 son la salida de la puerta OR, el complemento de la salida de la puerta OR, la salida Q del flip-flop y el complemento de la salida del flip-flop, $\overline{Q}$. Esto permite que la salida de la OLMC sea activa a nivel ALTO o a nivel BAJO, en cada modo. El multiplexor 1-de-2 conecta la salida del buffer de tres estados o la salida $\overline{Q}$ del flip-flop a través de un buffer a la matriz AND, en función del estado de S_1. Los bits de selección, S_0 y S_1 para cada OLMC, se programan en un grupo dedicado de celdas de la matriz mediante el software de compilación, por lo que el usuario no puede manipular directamente estos bits.

El flip-flop que se utiliza es tipo D disparado por flanco positivo. La entrada de reset asíncrona (*AR*) pone al flip-flop en el estado 0 lógico ($Q = 0$) independientemente de la señal de reloj. La entrada de inicialización síncrona (*SP*) activa el flip-flop poniéndolo en estado 1 ($Q = 1$), en el flanco de subida del impulso de reloj. La GAL22V10 tiene términos producto para las entradas *AR* y *SP*.

Las cuatro configuraciones de la OLMC son:

- Modo combinacional con salida activa a nivel BAJO
- Modo combinacional con salida activa a nivel ALTO
- Modo secuencial con salida activa a nivel BAJO
- Modo secuencial con salida activa a nivel ALTO

Los modos combinacionales se vieron en el Capítulo 7. Los modos secuenciales se cubren en la Sección 11.2.

La OLMC de la GAL16V8

Como se ha visto en el Capítulo 7, la OLMC de la GAL16V8 es algo diferente de la que dispone la GAL22V10. Una OLMC de la GAL16V8 se puede programar en uno de los tres modos disponibles para emular la mayoría de las PAL existentes, es decir, una GAL16V8 puede reemplazar a la PAL para la que se ha programado emular. Dispone de dos modos combinacionales, *simple* y *complejo*, y un modo secuencial.

Diagrama lógico. La Figura 11.3 muestra un diagrama lógico básico para la OLMC de la GAL16V8. Cada OLMC contiene dos **celdas globales**, $\overline{SYN}$ y $AC0$, que afectan a todas las OLMC. Existen también dos **celdas locales** para cada OLMC, $AC1(n)$ y XOR, que sólo afectan a la OLMC asociada. Las celdas globales y locales sirven para seleccionar varios caminos en la OLMC y determinar el modo combinacional o secuencial.

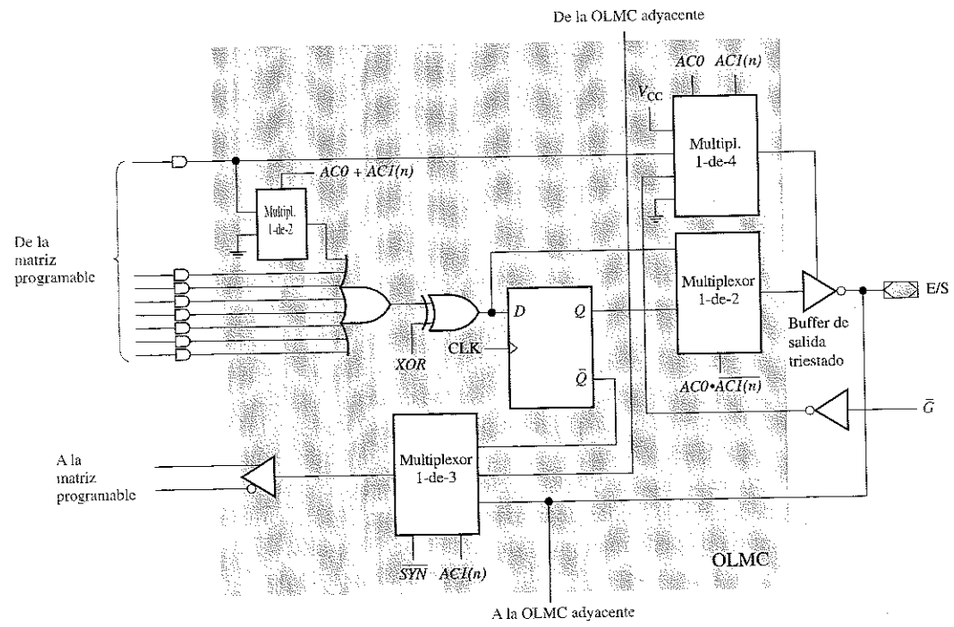


FIGURA 11.3
La OLMC de la GAL16V8.

Los bits $\overline{SYN}$ y $AC0$ controlan la configuración del modo. $\overline{SYN} = 0$ y $AC0 = 1$ establecen el modo secuencial, y $\overline{SYN} = 1$ y $AC0 = 0$ determinan los modos combinacionales. El bit $AC1$ controla la configuración de entrada/salida. El bit XOR selecciona la polaridad de salida, activa a nivel BAJO ($XOR = 0$) o activa a nivel ALTO ($XOR = 1$).

REVISIÓN DE LA SECCIÓN 11.1

1. ¿Cuáles son los dos modos de la GAL22V10?
2. ¿Cómo se determina el modo?
3. ¿Cuáles son los modos de la GAL16V8?

11.2 ■ SELECCIÓN DEL MODO DE LAS OLMC

En esta sección examinaremos la configuración de los modos de la GAL22V10 con fines ilustrativos y comentaremos las diferencias entre el modo secuencial y el modo combinacional. Al igual que en el Capítulo 7, se utiliza el lenguaje ABEL para ilustrar los aspectos software de las aplicaciones de los PLD.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Conocer el papel de los bits de control S_0 y S_1 en la selección de modo. ■ Mostrar cómo se pueden conseguir salidas activas a nivel BAJO y a nivel ALTO.

El modo combinacional

Los modos en la GAL22V10 se determinan mediante los bits S_0 y S_1, que se controlan mediante programación. En el modo combinacional, $S_1S_0 = 10$ o $S_1S_0 = 11$. La Figura 11.4 muestra los caminos lógicos a través de la OLMC para obtener una salida combinacional activa a nivel BAJO y a nivel ALTO, junto con el esquema lógico efectivo de la OLMC, para cada caso. En el modo combinacional, el flip-flop no se utiliza.

En la Figura 11.4(a), la inversión del buffer de salida triestado produce una salida activa a nivel BAJO. En la parte (b), la inversión del multiplexor 1-de-4 y la cancelación del buffer de salida triestado dan lugar a la salida activa a nivel ALTO.

El modo secuencial

En el modo secuencial, $S_1S_0 = 00$ o $S_1S_0 = 01$. La Figura 11.5 muestra los caminos lógicos a través de la OLMC que proporcionan salidas secuenciales activas a nivel BAJO y a nivel ALTO, junto con la lógica efectiva de la OLMC para cada caso. En la parte (a), la inversión del buffer de salida triestado produce una salida activa a nivel BAJO. En la parte (b), la inversión del flip-flop (salida $\overline{Q}$) y la cancelación del buffer de tres estados, dan lugar a una salida activa a nivel ALTO. Obsérvese que la realimentación a la matriz AND a través de un buffer se realiza a partir de la salida $\overline{Q}$ del flip-flop y no del pin de salida como en el caso del modo combinacional, por lo que una salida secuencial no puede utilizarse como entrada.

Especificación del modo por software

Los modos de una OLMC se establecen por software. Se hace por medio de las instrucciones específicas en la sección de declaraciones del archivo de entrada y en la forma en que se escriben las descripciones lógicas. Existen varios elementos básicos de las declaraciones y expresiones en ABEL que son importantes para programar una salida determinada como combinacional o secuencial.

1. La instrucción ISTYPE se usa para declarar una salida como combinacional o secuencial, mediante los atributos 'com' o 'reg', respectivamente. Por ejemplo, para especificar una salida dada como secuencial, se puede incluir una instrucción ISTYPE en una declaración de pines de la forma siguiente:

   ```
   Q0 PIN 23 ISTYPE 'reg';
   ```

   La salida también se puede especificar como no invertida o invertida (activa a nivel ALTO o activa a nivel BAJO) utilizando los atributos 'buffer' o 'invert' en la instrucción ISTYPE. Por ejemplo,

   ```
   Q0 PIN 23 ISTYPE 'reg,buffer';
   ```

   lo que indica que la salida secuencial Q_0 no está invertida. La numeración de los pines es opcional y si no se define, ABEL puede determinarla por el usuario.

2. Los operadores de asignación : = y :> se usan en las descripciones lógicas para indicar una salida secuencial. El operador de asignación secuencial : = es análogo al operador = , utilizado en las expresiones lógicas combinacionales. El operador de asignación secuencial :> es análogo al operador -> usado en las tablas de verdad de la lógica combinacional. Por ejemplo, la expresión de salida secuencial

   ```
   Q0 : = D0;
   ```

   indica que la salida Q_0 tomará el valor de la entrada D_0 con el impulso de reloj (en el caso de un flip-flop síncrono) y mantendrá dicho valor hasta el siguiente impulso de reloj. A modo de comparación, la expresión combinacional sería:

   ```
   X = A;
   ```

   lo que indica que la X de salida es igual a la A de entrada.

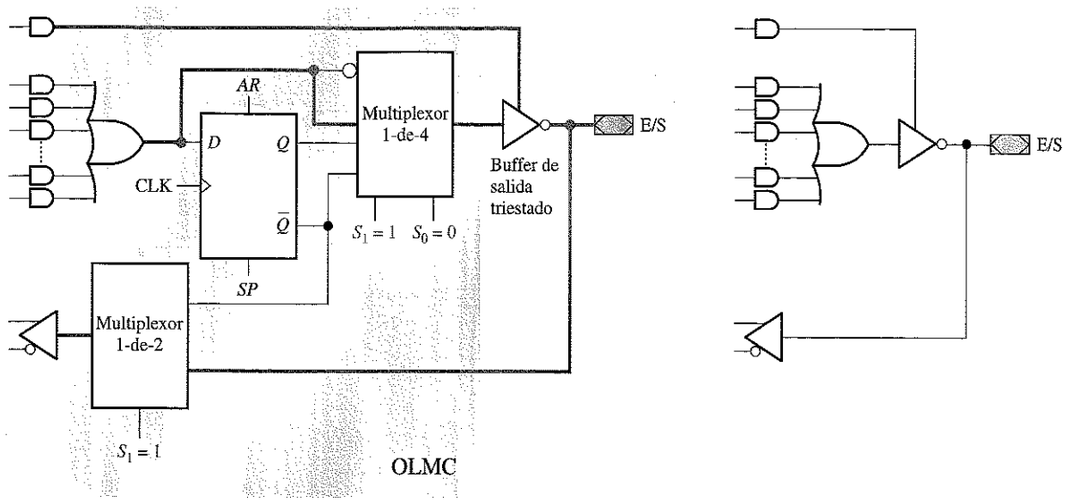
3. La extensión .CLK se usa para indicar que el dispositivo de registro es un flip-flop sincronizado con la señal de reloj. Una ecuación que utilice esta extensión de punto debe acompañar a la ecuación de salida. Por ejemplo, la ecuación de salida

   ```
   Q0 : = D0
   ```

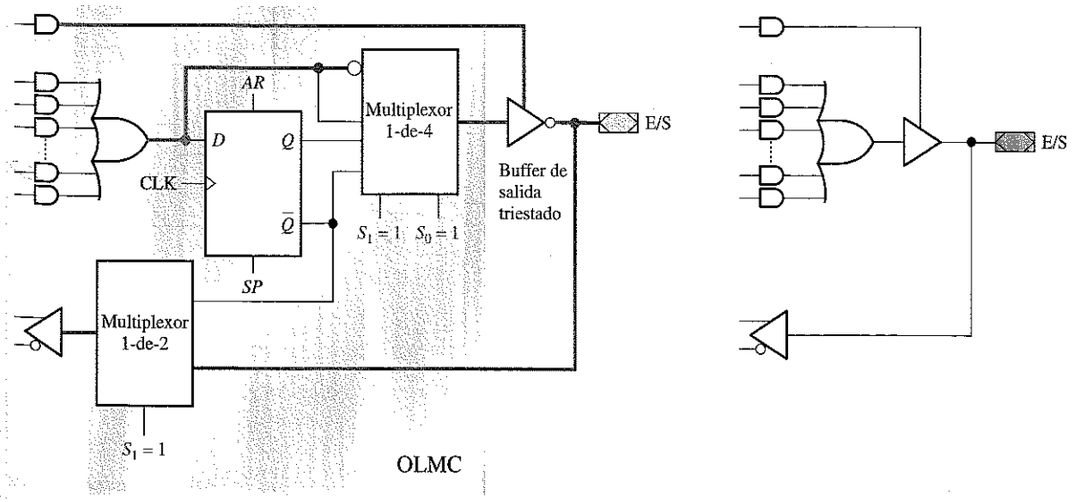
   debe ir seguida de una ecuación de reloj tal como la siguiente:

   ```
   Q0.CLK = Reloj;
   ```

   ABEL es un lenguaje extenso y los elementos mencionados representan sólo unos pocos de los muchos de los que dispone. Sin embargo, estos pocos son esenciales para programar un PLD que funcione como circuito secuencial.

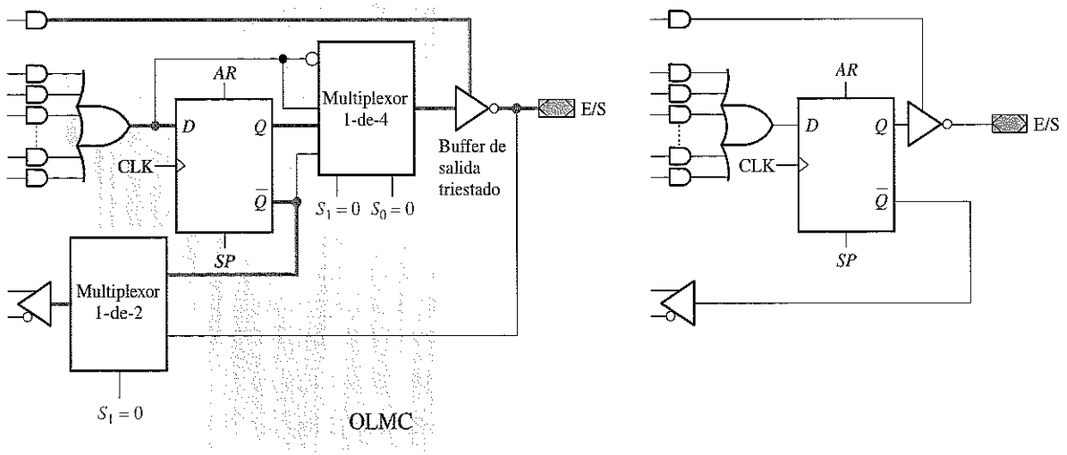


(a) OLMC en modo combinacional activo a nivel BAJO y diagrama lógico efectivo

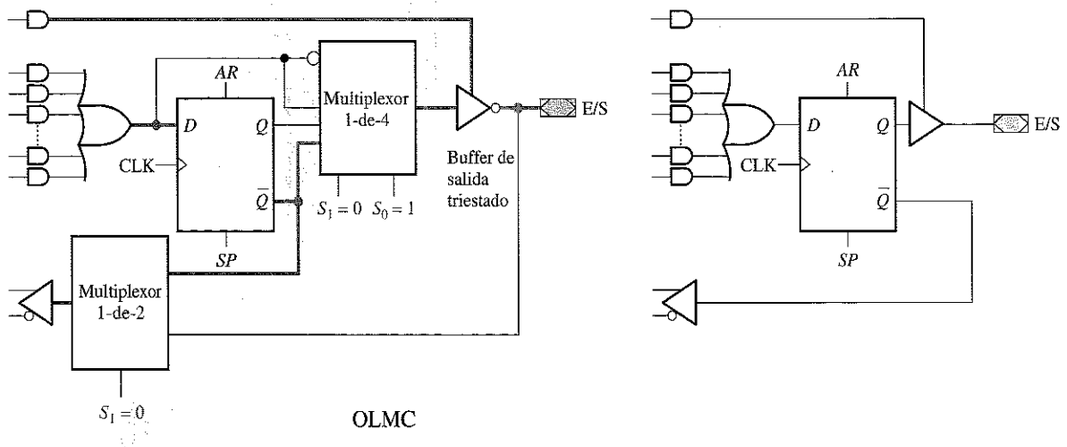


(b) OLMC en modo combinacional activo a nivel ALTO y diagrama lógico efectivo

FIGURA 11.4
Modo combinacional para salidas activas a nivel BAJO y a nivel ALTO. Las líneas más gruesas indican los caminos lógicos para cada caso.



(a) OLMC en modo combinacional activo a nivel BAJO y diagrama lógico efectivo



(b) OLMC en modo combinacional activo a nivel ALTO y diagrama lógico efectivo

FIGURA 11.5
Modo secuencial para salidas activas a nivel BAJO y a nivel ALTO. Las líneas más gruesas indican los caminos lógicos para cada caso.

EJEMPLO 11.1 Escribir en ABEL las ecuaciones y declaraciones de pines para especificar la GAL22V10 de la Figura 11.6 como un registro sincronizado con la señal de reloj con entrada y salida paralelo, siendo las entradas D_0 a D_7 y las salidas secuenciales no invertidas Q_0 a Q_7.

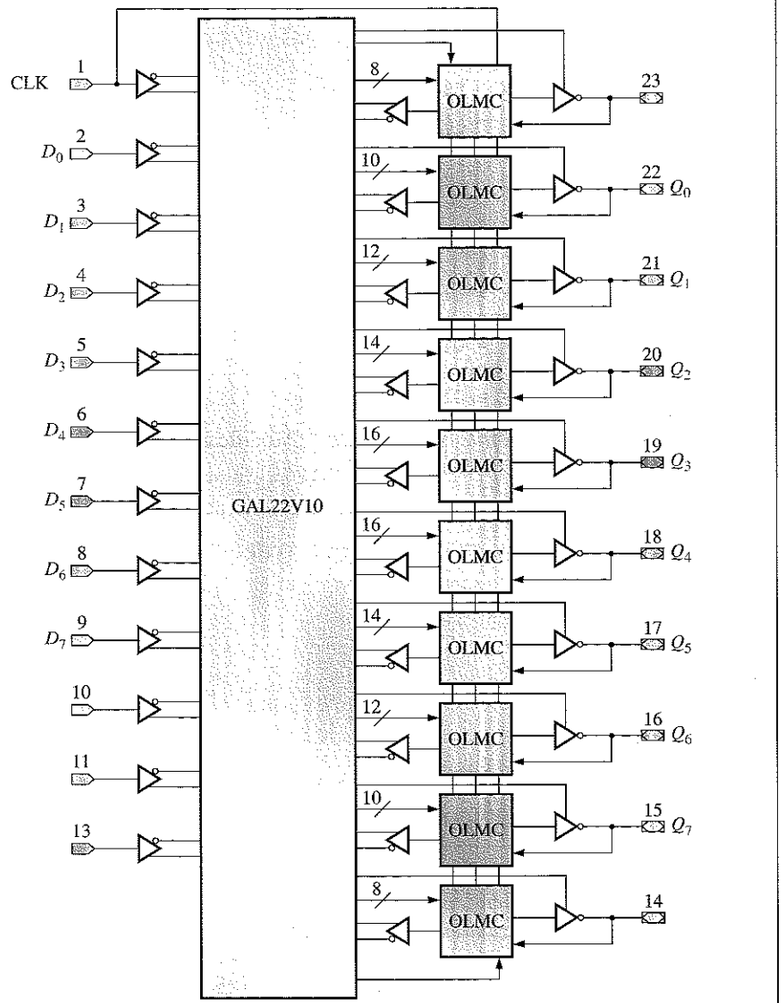


FIGURA 11.6

Solución. Las instrucciones de declaración de pines se pueden escribir del siguiente modo (la numeración de los pines es opcional):

```
Clk,D0,D1,D2,D3,D4,D5,D6,D7  PIN 1,2,3,4,5,6,7,8,9;
Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7  PIN 22,21,20,19,18,17,16,15 ISTYPE 'reg,buffer';
```

Las ecuaciones se pueden escribir utilizando la notación establecida, del siguiente modo

```
[Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7] := [D0, D1, D2, D3, D4, D5, D6, D7];
[Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7].CLK = Clk;
```

Esto es equivalente, aunque de forma mucho más sencilla, a escribir las ecuaciones individuales siguientes

```
Q0 := D0;
Q0.CLK = Clk;
Q1 := D1;
Q1.CLK = Clk;
Q2 := D2;
Q2.CLK = Clk;
Q3 := D3;
Q3.CLK = Clk;
Q4 := D4;
Q4.CLK = Clk;
Q5 := D5;
Q5.CLK = Clk;
Q6 := D6;
Q6.CLK = Clk;
Q7 := D7;
Q7.CLK = Clk;
```

Problema relacionado. Modificar las instrucciones de declaración de pines para obtener salidas invertidas.

REVISIÓN DE LA SECCIÓN 11.2

1. ¿Cuál es la diferencia básica entre el modo combinacional y el modo secuencial?
2. ¿Cómo puede utilizarse la instrucción ISTYPE?
3. ¿Cuál es la diferencia entre el operador : = y el operador = en una ecuación?
4. ¿Cuál es el propósito de la extensión .CLK?

11.3 ■ IMPLEMENTACIÓN DE REGISTROS DE DESPLAZAMIENTO CON PLD

En la última sección, aprenderemos cómo se usa el flip-flop de la OLMC en el modo secuencial. En esta sección, vamos a ver cómo se implementan los registros de desplazamiento con un PLD. De nuevo, utilizaremos la GAL22V10 con fines de ilustración.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Utilizar la extensión .CLK ■ Utilizar la extensión .AR ■ Utilizar las constantes especiales .c. y .x. en los vectores de prueba. ■ Implementar un registro de desplazamiento con entrada serie y salida paralelo de 8 bits. ■ Implementar un registro de desplazamiento con entrada y salida paralelo de 4 bits.

Registro de desplazamiento con entrada serie y salida paralelo de 8 bits

La Figura 11.7 muestra un registro de desplazamiento de 8 bits en el que los datos se introducen en serie. Cuando la entrada de habilitación (*enable*) de la puerta AND está a nivel ALTO, los datos se introducen sincronizadamente con la señal de reloj en el flip-flop situado más a la izquierda, y se desplazan de izquierda a derecha. Se puede

disponer de la salida Q de cada flip-flop. La entrada de borrado (*clear*) activa a nivel BAJO pone a cero a todos los flip-flops ($Q = 0$) independientemente de la señal de reloj.

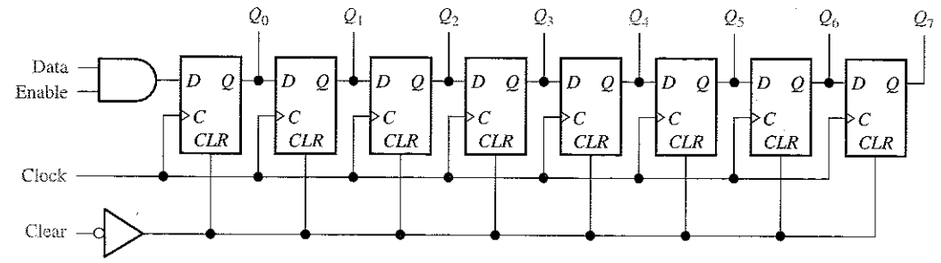


FIGURA 11.7
Registro de desplazamiento con entrada serie y salida paralelo de 8 bits

Implementación del registro de desplazamiento con un PLD

Las entradas y salidas del registro de desplazamiento se asignan a los pines de la GAL22V10, tal y como se muestra en la Figura 11.8. Puesto que la entrada D del primer flip-flop procede de la puerta AND, la salida Q_0 del mismo se expresa en ABEL mediante la ecuación secuencial siguiente:

```
Q0 := Data & Enable;
```

Puesto que la salida Q de cada flip-flop se conecta a la entrada D del siguiente flip-flop, las salidas de los restantes flip-flops se pueden expresar como sigue:

```
Q1 := Q0;
Q2 := Q1;
Q3 := Q2;
Q4 := Q3;
Q5 := Q4;
Q6 := Q5;
Q7 := Q6;
```

Desarrollo del archivo de entrada en ABEL

Recuérdese que, típicamente, el módulo del archivo de entrada en ABEL está formado por tres partes: declaraciones, descripciones lógicas y vectores de prueba. En el archivo de entrada para el registro de desplazamiento de 8 bits, la sección de declaraciones consiste en la asignación de pines para las entradas y salidas.

La sección de las ecuaciones expresa las ecuaciones lógicas de salida, utilizando la notación de conjuntos que se ha introducido en el Capítulo 7. En la ecuación Clock, la extensión .CLK especifica la entrada de reloj para cada flip-flop asociado con las salidas especificadas. En la ecuación Clear, la extensión .AR especifica una puesta a cero asíncrona para cada flip-flop asociado con las salidas especificadas.

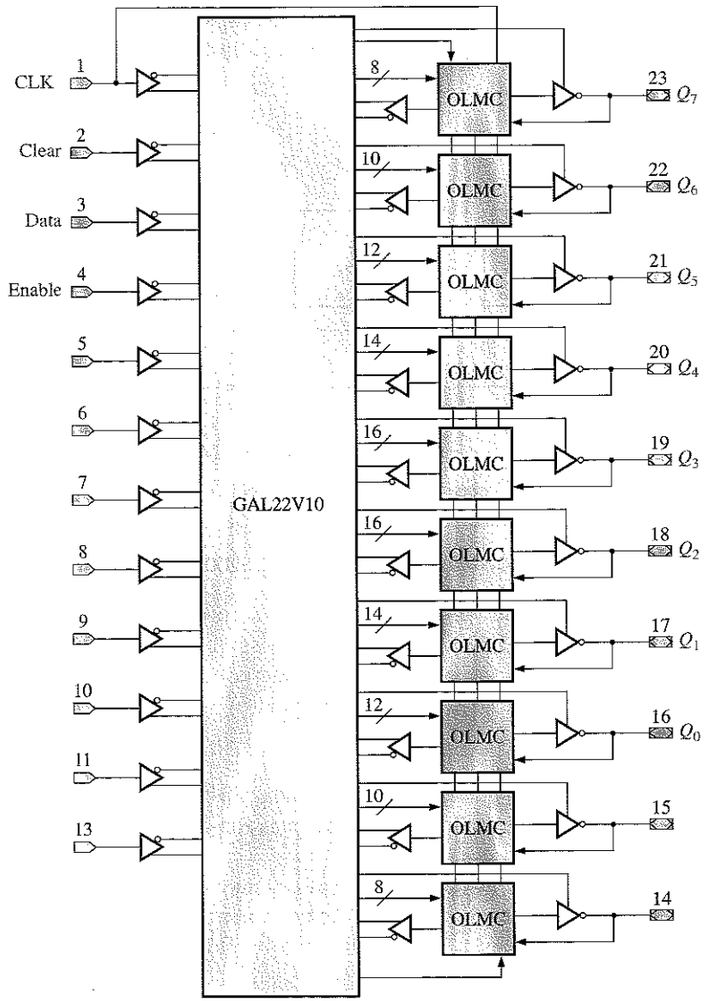


FIGURA 11.8
Asignación de entrada/salida de la GAL22V10 para el registro de desplazamiento de 8 bits.

Los elementos de la sección de los vectores de prueba constan de valores numéricos, la constante especial .c., que se usa para indicar el impulso de reloj y la constante .x., que indica una condición indiferente. Recuérdese que los vectores de prueba se usan para probar el PLD, para asegurarse de que funciona como se esperaba cuando se programó. El archivo de entrada en ABEL es el siguiente:

```
Module Registro_de_desplazamiento_de_8_bits

Title  'Registro de desplazamiento de 8 bits mediante GAL22V10'

"Declaración de dispositivo
```



```
   Registro  device  'P22V10'

   "Declaración de pines

   Clock, Clear              pin 1,2;
   Data, Enable              pin 3,4;
   Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7    pin 16,17,18,19,20,21,22,23 ISTYPE
                             'reg,buffer';

Equations

   Q0 := Data & Enable;

   [Q1,Q2,Q3,Q4,Q5,Q6,Q7] := [Q0,Q1,Q2,Q3,Q4,Q5,Q6];

   [Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7].CLK = Clock;

   [Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7].AR = !Clear;

Test_Vectors

   ([Clock, Clear, Data, Enable] -> [Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7])
    [ .x. ,   0   ,  .x. ,   .x.  ] -> [ 0, 0, 0, 0, 0, 0, 0, 0];
    [ .c. ,   1   ,   1  ,    0   ] -> [ 0, 0, 0, 0, 0, 0, 0, 0];
    [ .c. ,   1   ,   0  ,    1   ] -> [ 0, 0, 0, 0, 0, 0, 0, 0];
    [ .c. ,   1   ,   1  ,    1   ] -> [ 1, 0, 0, 0, 0, 0, 0, 0];
    [ .c. ,   1   ,   0  ,    1   ] -> [ 0, 1, 0, 0, 0, 0, 0, 0];
    [ .c. ,   1   ,   1  ,    1   ] -> [ 1, 0, 1, 0, 0, 0, 0, 0];
    [ .c. ,   1   ,   0  ,    1   ] -> [ 0, 1, 0, 1, 0, 0, 0, 0];
    [ .c. ,   1   ,   1  ,    1   ] -> [ 1, 0, 1, 0, 1, 0, 0, 0];
    [ .c. ,   1   ,   0  ,    1   ] -> [ 0, 1, 0, 1, 0, 1, 0, 0];
    [ .c. ,   1   ,   1  ,    1   ] -> [ 1, 0, 1, 0, 1, 0, 1, 0];
    [ .c. ,   1   ,   0  ,    1   ] -> [ 0, 1, 0, 1, 0, 1, 0, 1];
    [ .c. ,   1   ,   1  ,    1   ] -> [ 1, 0, 1, 0, 1, 0, 1, 0];
    [ .c. ,   0   ,   1  ,    1   ] -> [ 0, 0, 0, 0, 0, 0, 0, 0];

END
```

EJEMPLO 11.2 Implementar el registro de desplazamiento de 4 bits con entrada paralelo y salida serie de la Figura 11.9, desarrollando un archivo de entrada en ABEL para la GAL22V10. La entrada *SH/$\overline{LD}$* corresponde a Shift/$\overline{\text{Load}}$ (desplazamiento/carga). Cuando esta entrada está a nivel BAJO, los datos paralelo se cargan en el registro, y cuando está a nivel ALTO, los datos se desplazan de izquierda a derecha. D_0 a D_3 son las entradas de datos paralelo y Q_3 es la salida de datos serie.

Solución. El archivo de entrada en ABEL es el siguiente:

```
Module  Registro_de_desplazamiento_de_4_bits

Title  'Registro de desplazamiento de 4 bits mediante
       GAL22V10'

"Declaración de dispositivo

   Registro  device  'P22V10'

"Declaración de pines

   Clock, Clear      pin 1,2;
   SHLD              pin 3;
   D0, D1, D2, D3    pin 4,5,6,7 ISTYPE 'reg,buffer';
   Q0, Q1, Q2, Q3    pin 14,15,16,17 ISTYPE 'reg,buffer';
```

```
Equations
   Q0 := D0;
   Q1 := Q0 & SHLD # D1 & !SHLD;
   Q2 := Q1 & SHLD # D2 & !SHLD;
   Q3 := Q2 & SHLD # D3 & !SHLD;
   [Q0, Q1, Q2, Q3].CLK = Clock;
   [Q0, Q1, Q2, Q3].AR = !Clear;
Test_Vectors
   ([Clock, Clear, SHLD,  D0 ,  D1 ,  D2 ,  D3 ] -> [Q3])
    [ .x. ,   0   , .x. , .x., .x., .x., .x.] -> [ 0];
    [ .c. ,   1   ,  0  ,  0 ,  1 ,  0 ,  1 ] -> [ 1];
    [ .c. ,   1   ,  0  ,  1 ,  0 ,  1 ,  0 ] -> [ 0];
    [ .c. ,   1   ,  1  ,  1 ,  0 ,  1 ,  0 ] -> [ 1];
    [ .c. ,   1   ,  1  ,  1 ,  0 ,  1 ,  0 ] -> [ 0];
    [ .c. ,   1   ,  1  ,  1 ,  0 ,  1 ,  0 ] -> [ 1];
    [ .c. ,   0   ,  0  ,  1 ,  0 ,  1 ,  0 ] -> [ 0];
END
```

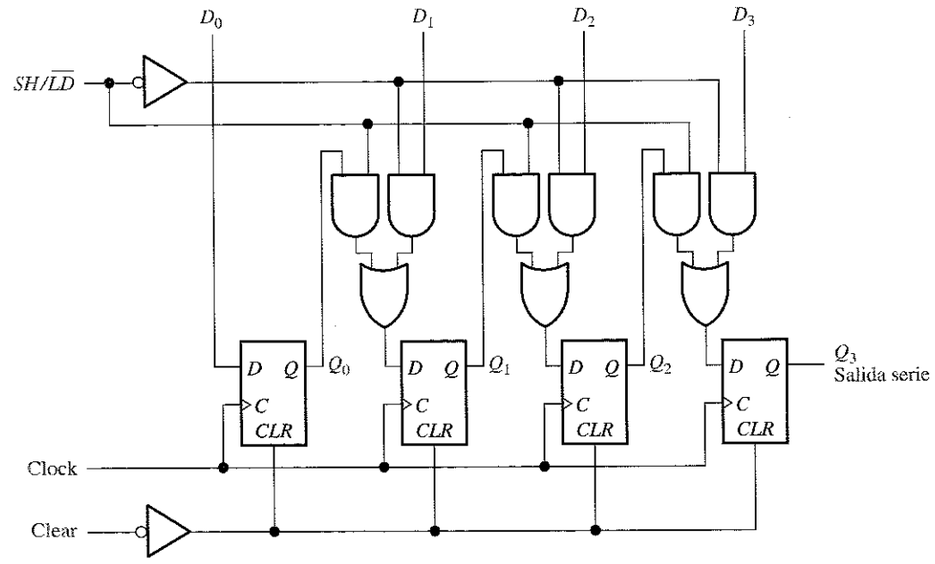


FIGURA 11.9
Registro de desplazamiento con entrada paralelo y salida serie de 4 bits.

Problema relacionado. Modificar las secciones de declaración de pines y ecuaciones del archivo de entrada en ABEL, para que el registro de desplazamiento de 4 bits de la Figura 11.9 sea de cinco bits.

REVISIÓN DE LA SECCIÓN 11.3

1. ¿Cuál es el significado de la ecuación ABEL Q0 := A & !B # !A & B;?
2. ¿Cuál es el propósito de la extensión .AR?
3. ¿Cuándo se utiliza la constante especial .c.?

11.4 ■ IMPLEMENTACIÓN DE CONTADORES CON PLD

Como ya sabe, los contadores están formados por flip-flops al igual que los registros de desplazamiento, pero los contadores siguen una secuencia de estados prescrita, mientras que los registros no. Los contadores pertenecen a una categoría de circuitos digitales, conocida como máquinas de estado, como ya se mencionó brevemente en el Capítulo 9. En esta sección, se estudian los tipos generales de máquinas de estado y se cubren tres métodos para implementar un contador con un PLD.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Definir las dos categorías de máquinas de estado. ■ Implementar un contador mediante ecuaciones. ■ Implementar un contador mediante las tablas de verdad. ■ Implementar un contador mediante un diagrama de estados.

Tipos de máquinas de estados

Los circuitos lógicos secuenciales se pueden clasificar dentro de la categoría de circuitos conocidos como **máquinas de estados**, de las que existen dos tipos básicos. En la máquina de estados de Moore que se muestra en la Figura 11.10(a), la salida depende sólo del estado interno y de cualquier entrada sincronizada con el circuito. Ejemplos de las máquinas de estados de tipo Moore son los contadores, en las que los flip-flops son los elementos de memoria.

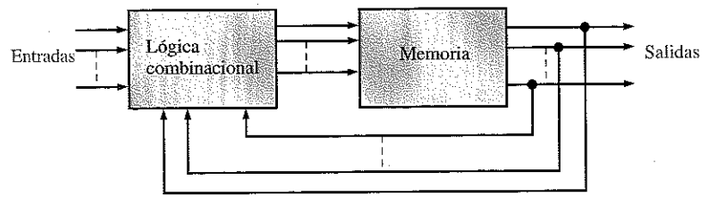
En la máquina de estados Mealy, las salidas vienen determinadas por el estado interno y por entradas que no están sincronizadas con el circuito, como se indica en la Figura 11.10(b).

Diseño de un contador en código Gray ascendente/descendente de 3 bits

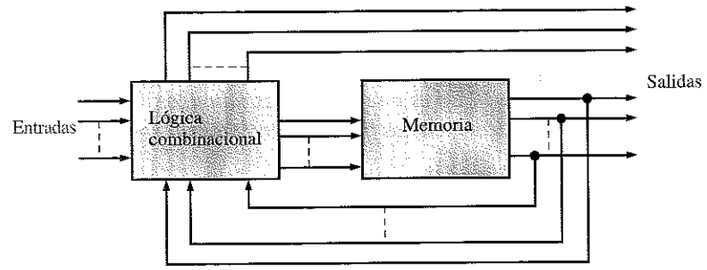
Para ilustrar el proceso de implementación de un contador en un PLD, se va a utilizar un contador en código Gray ascendente/descendente de 3 bits.

El primer paso consiste en diseñar el contador con flip-flops D. Se utiliza este tipo de flip-flop porque la mayoría de los PLD, incluyendo la GAL22V10 y la GAL16V8, disponen de este tipo de flip-flops en sus macroceldas OLMC. En el Capítulo 9 se diseñó un contador similar con flip-flops J-K.

El diagrama de estados. La Figura 11.11 muestra el diagrama de estados para un contador en código Gray ascendente/descendente de 3 bits. Las expresiones $Y = 1$ o $Y = 0$ que haya junto a cada flecha indican el estado de la entrada de control ascendente/descendente.



(a) Máquina de estados Moore



(b) Máquina de estados Mealy

FIGURA 11.10
Diagramas de bloques generales para los dos tipos de máquinas de estados.

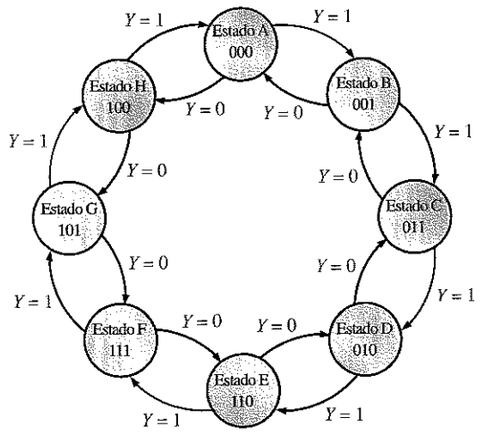


FIGURA 11.11
Diagrama de estados para el contador en código Gray ascendente/descendente de 3 bits.

Tabla del estado futuro. La tabla del estado futuro se deriva del diagrama de estados, y se muestra en la Tabla 11.1. Para cada estado actual existen dos posibles estados siguientes, dependiendo de la variable de control UP/DOWN (ascendente/descendente), *Y*.

TABLA 11.1
Tabla del estado siguiente del contador en código Gray ascendente/descendente de 3 bits.

Estado actual			Estado siguiente Y = 0 (Descendente)			Estado siguiente Y = 1 (Ascendente)		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1	1
0	1	1	0	0	1	0	1	0
0	1	0	0	1	1	1	1	0
1	1	0	0	1	0	1	1	1
1	1	1	1	1	0	1	0	1
1	0	1	1	1	1	1	0	0
1	0	0	1	0	1	0	0	0

Tabla de transiciones del flip-flop. La tabla de transiciones para el flip-flop D se muestra en la Tabla 11.2. Esta tabla especifica la entrada *D* requerida para cada una de las cuatro posibles transiciones de estado. Q_N es el estado actual y Q_{N+1} es el estado siguiente.

TABLA 11.2
Tabla de transiciones de un flip-flop D.

Transiciones de salida		Entrada del flip-flop
Q_N	Q_{N+1}	*D*
0 ⟶	0	0
0 ⟶	1	1
1 ⟶	0	0
1 ⟶	1	1

Mapas de Karnaugh. Las entradas *D* de los tres flip-flops del contador en código Gray se transfieren a los mapas de Karnaugh como muestra la Figura 11.12. La entrada de control up/down *Y* se trata como una de las variables de estado, junto con Q_0, Q_1 y Q_2. Utilizando la tabla del estado siguiente (Tabla 11.1), se transfieren las entradas de los flip-flops de la Tabla 11.2 a los mapas para cada estado actual del contador.

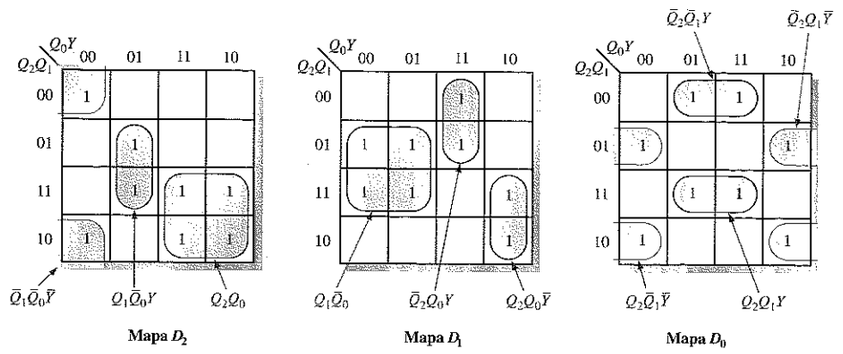


FIGURA 11.12
Mapas de la entrada D para el contador en código Gray ascendente/descendente de 3 bits.

El circuito contador. La Figura 11.13 muestra el diagrama lógico del contador en código Gray ascendente/descente de 3 bits. La lógica combinacional se emplea para implementar las tres ecuaciones correspondientes a las entradas D de los flip-flops.

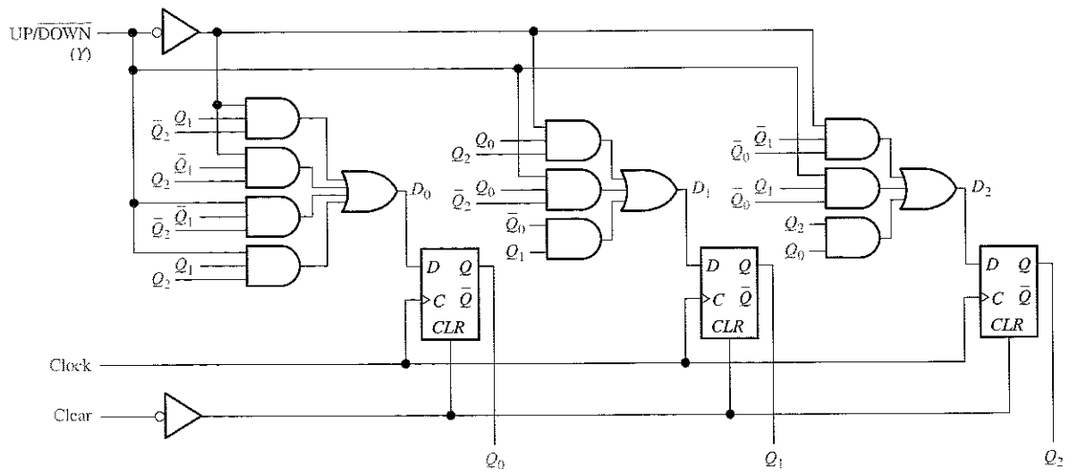


FIGURA 11.13
Diagrama lógico del contador en código Gray ascendente/descendente de 3 bits.

Expresiones booleanas para las entradas D

Los 1s de los mapas se combinan para formar los grupos más grandes posibles, como se ilustra en la Figura 11.12, y dichos grupos se expresan como productos dando lugar a las siguientes expresiones:

$$D_0 = Q_2Q_1Y + \overline{Q}_2\overline{Q}_1Y + Q_2\overline{Q}_1\overline{Y} + \overline{Q}_2Q_1\overline{Y}$$
$$D_1 = Q_2Q_0\overline{Y} + \overline{Q}_2Q_0Y + Q_1\overline{Q}_0$$
$$D_2 = \overline{Q}_1\overline{Q}_0\overline{Y} + Q_1\overline{Q}_0Y + Q_2Q_0$$

Implementación del contador con ecuaciones ABEL

Ya estamos familiarizados con el uso de las ecuaciones ABEL para describir un circuito secuencial con salidas secuenciales. El proceso para los contadores es básicamente el mismo que para los registros de desplazamiento, en cuanto a estructura del archivo de entrada en ABEL, como se verá en el Ejemplo 11.3.

Además de la constante especial .c. para un impulso de reloj, se puede utilizar otra constante especial .x., en los vectores de prueba y tablas de verdad, para indicar una condición indiferente en una entrada o en una salida.

EJEMPLO 11.3 Desarrollar un archivo de entrada en ABEL que utilice el formato de las ecuaciones lógicas, para implementar el contador en código Gray ascendente/descendente de 3 bits de la Figura 11.13. El dispositivo objeto es la GAL22V10.

Solución. Y representa la entrada de control $UP/\overline{DOWN}$. Puesto que la salida Q toma el valor de la entrada D en cada impulso de reloj, la ecuación en ABEL se expresa en función de las salidas Q en lugar de las entradas D. El archivo de entrada es el siguiente:

```
Module Contador_3_bits_up/down_código_Gray
Title 'Contador up/down de 3 bits en código Gray mediante GAL22V10'
"Declaración de dispositivo
       Contador  device  'P22V10'
"Declaración de pines
       Clock, Clear    pin 1,2;
       Y               pin 3;
       Q0,Q1,Q2        pin 21,22,23 ISTYPE 'reg,buffer';
   Equations
       Q0 := Q2 & Q1 & Y # !Q2 & !Q1 & Y # Q2 & !Q1 & !Y # !Q2 &
             Q1 & !Y;
       Q1 := Q2 & Q0 & !Y # !Q2 & Q0 & Y # Q1 & !Q0;
       Q2 := !Q1 & !Q0 & !Y # Q1 & !Q0 & Y # Q2 & Q0;
       [Q0,Q1,Q2].CLK = Clock;
       [Q0,Q1,Q2].AR = !Clear;
   Test_Vectors ([Clock, Clear,  Y ] -> [Q2,Q1,Q0])
                 [ .c. ,   0  , .x.] -> [ 0, 0, 0];
                 [ .c. ,   1  ,  0 ] -> [ 1, 0, 0];
                 [ .c. ,   1  ,  0 ] -> [ 1, 0, 1];
                 [ .c. ,   1  ,  0 ] -> [ 1, 1, 1];
                 [ .c. ,   1  ,  0 ] -> [ 1, 1, 0];
                 [ .c. ,   1  ,  0 ] -> [ 0, 1, 0];
                 [ .c. ,   1  ,  0 ] -> [ 0, 1, 1];
                 [ .c. ,   1  ,  0 ] -> [ 0, 0, 1];
                 [ .c. ,   1  ,  0 ] -> [ 0, 0, 0];
                 [ .c. ,   1  ,  1 ] -> [ 0, 0, 1];
                 [ .c. ,   1  ,  1 ] -> [ 0, 1, 1];
                 [ .c. ,   1  ,  1 ] -> [ 0, 1, 0];
                 [ .c. ,   1  ,  1 ] -> [ 1, 1, 0];
                 [ .c. ,   1  ,  1 ] -> [ 1, 1, 1];
                 [ .c. ,   1  ,  1 ] -> [ 1, 0, 1];
                 [ .c. ,   1  ,  1 ] -> [ 1, 0, 0];
                 [ .c. ,   1  ,  1 ] -> [ 0, 0, 0];
END
```

Problema relacionado. Explicar, línea por línea, qué indican los vectores de prueba de este archivo de entrada.

Implementación del contador con una tabla de verdad ABEL

En lugar de utilizar las ecuaciones, el contador puede describirse utilizando el método de entrada de la tabla de verdad, que se cubrió en la Sección 7.7. El método de entrada que se utilice es, básicamente, una cuestión de preferencia personal.

EJEMPLO 11.4 Desarrollar un archivo de entrada en ABEL utilizando el formato de tabla de verdad para implementar el contador en código Gray ascendente/descendente de 3 bits de la Figura 11.13. El dispositivo objeto es una GAL22V10.

Solución. En una tabla de verdad, las entradas se sitúan a la izquierda del operador de asignación y las salidas a la derecha. El operador de asignación :> se usa en las tablas de verdad para circuitos con salidas secuenciales e indica que, para los valores de entrada dados, las salidas tomarán el estado determinado después del siguiente impulso de reloj. El archivo de entrada ABEL se indica a continuación. Obsérvese que contiene las ecuaciones que definen las entradas Clock y Clear.

```
Module  Contador_3_bits_up/down_código_Gray_

Title   'Contador up/down de 3 bits en código Gray mediante
        GAL22V10'

"Declaración de dispositivos
   Contador  device  'P22V10'

"Declaración de pines
   Clock, Clear    pin 1,2;
   Y               pin 3;
   Q0,Q1,Q2        pin 21,22,23 ISTYPE 'reg,buffer';

Equations

   [Q0,Q1,Q2].CLK = Clock;

   [Q0,Q1,Q2].AR = !Clear;

Truth_Table ([Clear,  Y  ,  Q2  ,  Q1  ,  Q0 ] :> [Q2, Q1, Q0])
             [  0   , .x. , .x. , .x. , .x.] :> [ 0,  0,  0];
             [  1   ,  0  ,  0  ,  0  ,  0 ] :> [ 1,  0,  0];
             [  1   ,  0  ,  1  ,  0  ,  0 ] :> [ 1,  0,  1];
             [  1   ,  0  ,  1  ,  0  ,  1 ] :> [ 1,  1,  1];
             [  1   ,  0  ,  1  ,  1  ,  1 ] :> [ 1,  1,  0];
             [  1   ,  0  ,  1  ,  1  ,  0 ] :> [ 0,  1,  0];
             [  1   ,  0  ,  0  ,  1  ,  0 ] :> [ 0,  1,  1];
             [  1   ,  0  ,  0  ,  1  ,  1 ] :> [ 0,  0,  1];
             [  1   ,  0  ,  0  ,  0  ,  1 ] :> [ 0,  0,  0];
             [  1   ,  1  ,  0  ,  0  ,  0 ] :> [ 0,  0,  1];
             [  1   ,  1  ,  0  ,  0  ,  1 ] :> [ 0,  1,  1];
             [  1   ,  1  ,  0  ,  1  ,  1 ] :> [ 0,  1,  0];
             [  1   ,  1  ,  0  ,  1  ,  0 ] :> [ 1,  1,  0];
             [  1   ,  1  ,  1  ,  1  ,  0 ] :> [ 1,  1,  1];
             [  1   ,  1  ,  1  ,  1  ,  1 ] :> [ 1,  0,  1];
             [  1   ,  1  ,  1  ,  0  ,  1 ] :> [ 1,  0,  0];
             [  1   ,  1  ,  1  ,  0  ,  0 ] :> [ 0,  0,  0];

Test_Vectors ([Clock, Clear,  Y ] -> [Q2,Q1,Q0])
              [ .c. ,   0  , .x.] -> [ 0, 0, 0];
              [ .c. ,   1  ,  0 ] -> [ 1, 0, 0];
              [ .c. ,   1  ,  0 ] -> [ 1, 0, 1];
              [ .c. ,   1  ,  0 ] -> [ 1, 1, 1];
              [ .c. ,   1  ,  0 ] -> [ 1, 1, 0];
              [ .c. ,   1  ,  0 ] -> [ 0, 1, 0];
              [ .c. ,   1  ,  0 ] -> [ 0, 1, 1];
              [ .c. ,   1  ,  0 ] -> [ 0, 0, 1];
              [ .c. ,   1  ,  0 ] -> [ 0, 0, 0];
              [ .c. ,   1  ,  1 ] -> [ 0, 0, 1];
              [ .c. ,   1  ,  1 ] -> [ 0, 1, 1];
              [ .c. ,   1  ,  1 ] -> [ 0, 1, 0];
              [ .c. ,   1  ,  1 ] -> [ 1, 1, 0];
              [ .c. ,   1  ,  1 ] -> [ 1, 1, 1];
              [ .c. ,   1  ,  1 ] -> [ 1, 0, 1];
              [ .c. ,   1  ,  1 ] -> [ 1, 0, 0];
              [ .c. ,   1  ,  1 ] -> [ 0, 0, 0];

END
```

Problema relacionado. Explicar la utilización de la constante especial indiferente en la primera línea de la tabla de verdad.

Implementación del contador con un diagrama de estados ABEL

Una tercera alternativa para la introducción del diseño en ABEL es el método del diagrama de estado, que a menudo es mejor para las máquinas de estado más complejas. No obstante, el contador de 3 bits servirá para presentar este método.

El diagrama de estados del contador en código Gray ascendente/descendente de 3 bits se presentó en la Figura 11.11. Existen ocho estados etiquetados con las letras A a H, que representan el código Gray de tres estados. En cada estado, el contador puede pasar al estado siguiente inmediato superior o inmediato inferior, dependiendo del valor de la variable de control UP/$\overline{\text{DOWN}}$, *Y*.

Instrucciones IF-THEN-ELSE. ABEL proporciona una instrucción condicional para describir a qué estado pasará el contador en función de ciertas condiciones. En este caso, existen dos condiciones, $Y = 0$ e $Y = 1$. Por ejemplo, cuando el contador está en el estado A (000), pasará al estado B (001) si $Y = 1$ y, en caso contrario, pasará al estado H ($Y = 0$). La instrucción IF-THEN-ELSE de ABEL para el estado A es:

```
IF Y THEN B ELSE H;
```

Las instrucciones también pueden escribirse en minúsculas.

Instrucción WITH. Esta instrucción puede utilizarse en las ecuaciones para especificar salidas asociadas con transiciones específicas o en ecuaciones para salidas secuenciales, que deben activarse en un estado de destino.

Definición de estados. Cuando se usa el diagrama de estados, se debe incluir una definición de todos los estados del contador. En primer lugar, se define la variable *QESTADO* como un conjunto de salidas *Q*, que son los estados de los flip-flops:

```
QESTADO = [Q2, Q1, Q0];
```

Y, a continuación, se indican los nombres y valores para cada uno de los ocho estados.

```
A = [0 , 0 , 0];
B = [0 , 0 , 1];
C = [0 , 1 , 1];
D = [0 , 1 , 0];
E = [1 , 1 , 0];
F = [1 , 1 , 1];
G = [1 , 0 , 1];
H = [1 , 0 , 0];
```

El siguiente ejemplo muestra cómo se usan estos elementos básicos del diagrama de estados ABEL para describir el contador.

EJEMPLO 11.5 Desarrollar un archivo de entrada ABEL que utilice el diagrama de estados para implementar el contador en código Gray ascendente/descendente de 3 bits de la Figura 11.13. El dispositivo objeto es una GAL22V10.

Solución. El archivo de entrada ABEL es el siguiente. Obsérvese que se incluyen de nuevo las ecuaciones que definen las entradas CLK y Clear. Los vectores de prueba tampoco se muestran en este caso, ya que son los mismos del ejemplo 11.3.

```
Module    Contador_3_bits_up/down_código_Gray
Title     'Contador up/down de 3 bits en código Gray mediante GAL22V10'
"Declaración de dispositivo
          Contador  device  'P22V10'
"Declaración de pines
          Clock, Clear    pin 1,2;
          Y               pin 3;
          Q0,Q1,Q2        pin 21, 22, 23 ISTYPE 'reg,buffer';
"Definiciones de estados
          QESTADO  = [Q2,Q1,Q0];
          A        = [ 0, 0, 0];
          B        = [ 0, 0, 1];
          C        = [ 0, 1, 1];
          D        = [ 0, 1, 0];
          E        = [ 1, 1, 0];
          F        = [ 1, 1, 1];
          G        = [ 1, 0, 1];
          H        = [ 1, 0, 0];
Equations
          QESTADO.CLK = Clock;
          QESTADO.AR = !Clear;
State_diagram QESTADO
          State A: if Y then B else H;
          State B: if Y then C else A;
          State C: if Y then D else B;
          State D: if Y then E else C;
          State E: if Y then F else D;
          State F: if Y then G else E;
          State G: if Y then H else F;
          State H: if Y then A else G;
Test_Vectors ([Clock ,Clear,   Y  ] -> [Q2,Q1,Q0])
              [ .c.  ,  0  ,  .x. ] -> [ 0, 0, 0];
              [ .c.  ,  1  ,   0  ] -> [ 1, 0, 0];
              [ .c.  ,  1  ,   0  ] -> [ 1, 0, 1];
              [ .c.  ,  1  ,   0  ] -> [ 1, 1, 1];
              [ .c.  ,  1  ,   0  ] -> [ 1, 1, 0];
              [ .c.  ,  1  ,   0  ] -> [ 0, 1, 0];
              [ .c.  ,  1  ,   0  ] -> [ 0, 1, 1];
              [ .c.  ,  1  ,   0  ] -> [ 0, 0, 1];
              [ .c.  ,  1  ,   0  ] -> [ 0, 0, 0];
              [ .c.  ,  1  ,   1  ] -> [ 0, 0, 1];
              [ .c.  ,  1  ,   1  ] -> [ 0, 1, 1];
              [ .c.  ,  1  ,   1  ] -> [ 0, 1, 0];
              [ .c.  ,  1  ,   1  ] -> [ 1, 1, 0];
              [ .c.  ,  1  ,   1  ] -> [ 1, 1, 1];
              [ .c.  ,  1  ,   1  ] -> [ 1, 0, 1];
              [ .c.  ,  1  ,   1  ] -> [ 1, 0, 0];
              [ .c.  ,  1  ,   1  ] -> [ 0, 0, 0];
END
```

Problema relacionado. Escribir la instrucción IF-THEN-ELSE para el estado A, utilizando $\overline{Y}$ en lugar de Y.

Instrucción GOTO. Además de la instrucción IF-THEN-ELSE para transiciones de estado condicionales, ABEL proporciona la instrucción GOTO para transiciones de estado incondicionales. Si un contador sólo pasa a un estado siguiente a partir de un determinado estado, se puede utilizar una instrucción GOTO. Por ejemplo, un contador de 3 bits que pasa a través de una secuencia de cuenta binaria se puede describir mediante las definiciones de estado ABEL y el diagrama de estados siguientes:

```
QESTADO = [Q2,Q1,Q0];
A       = [ 0, 0, 0];
B       = [ 0, 0, 1];
C       = [ 0, 1, 0];
D       = [ 0, 1, 1];
E       = [ 1, 0, 0];
F       = [ 1, 0, 1];
G       = [ 1, 1, 0];
H       = [ 1, 1, 1];

State_diagram QESTADO

     State A:    GOTO B;
     State B:    GOTO C;
     State C:    GOTO D;
     State D:    GOTO E;
     State E:    GOTO F;
     State F:    GOTO G;
     State G:    GOTO H;
     State H:    GOTO A;
```

REVISIÓN DE LA SECCIÓN 11.4

1. Nombrar las dos categorías de máquinas de estados.
2. ¿Dentro de qué categoría de las máquinas de estados se clasifican los contadores?
3. Enumerar tres métodos de entrada ABEL que puedan utilizarse para describir una máquina de estados.
4. ¿Qué representa la constante especial .x.?

11.5 ■ IMPLEMENTACIÓN DE SISTEMAS MEDIANTE PLD

En esta sección se utiliza un ejemplo específico para ilustrar el uso de los diagramas de estado y otros métodos de entrada para implementar un sistema completo con PLD utilizando el lenguaje ABEL. El sistema ejemplo es un control de ascensor para un edificio de dos pisos y se utilizan dos GAL16V8. Este sistema se implementa a partir de una descripción de los requisitos del sistema, un esquema de bloques y un diagrama de estados.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir un sistema de control de ascensor. ■ Implementar un diseño a partir de un diagrama de estados. ■ Utilizar la extensión .FB

El sistema

En la Figura 11.14 se muestra un diagrama de bloques de un sistema de control de un ascensor para dos pisos. El sistema de control supervisa el estado de los pulsadores del ascensor, responde a las solicitudes de servicio y presenta en el display el número de piso y la dirección del ascensor. El sistema utiliza dos GAL16V8, una para la lógica de control y otra para la lógica del display. La cantidad de lógica necesaria podría implementarse con una única GAL. Sin embargo, una GAL16V8 no tiene bastantes salidas para acomodar los requisitos del sistema.

Existen dos pulsadores de "demanda" (LLAMAR 1 y LLAMAR 2) en cada piso al lado de la puerta del ascensor. Cuando el ascensor llega al piso solicitado, las puertas se abren durante un periodo de tiempo para que los pasajeros entren y salgan y luego, automáticamente, se cierran. Dentro del ascensor, se puede presionar el pulsador PISO2 o PISO1. Si se encuentra en el primer piso, se presiona el pulsador PISO2 para ir a dicho piso; y si se encuentra en el segundo piso, se presiona el pulsador PISO1 para ir al mismo. Si se presiona el pulsador de abertura del interior del ascensor, las puertas se volverán a abrir o permanecerán abiertas más tiempo que el tiempo preestablecido, siempre que el ascensor no se mueva. Si no se activa ninguno de los pulsadores, el ascensor permanece en el piso que se encuentre. Una señal de reloj muy lenta controla tanto los dos movimientos del ascensor como el tiempo que la puerta está abierta cuando se para en un piso.

La lógica de control de estados está formada por latches independientes de la señal de reloj, que almacenan el cierre momentáneo de los pulsadores, y una máquina de estados que genera las tres salidas de control del ascensor sincronizadas con el reloj: *PUERTA, MOVIM (movimiento)* y *DIR (dirección)*. La Tabla 11.3 muestra los estados de estas salidas sincronizadas.

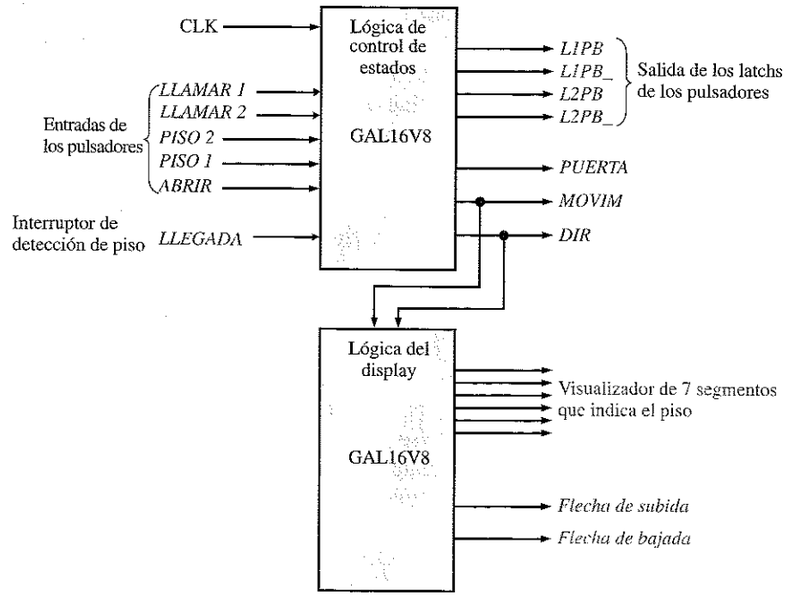


FIGURA 11.14
Diagrama de bloques del sistema de control del ascensor.

TABLA 11.3

Variable de salida	Estado	
PUERTA	0 = Abierta	1 = Cerrada
MOVIM	0 = En espera	1 = En movimiento
DIR	0 = Arriba	1= Abajo

La unidad del display es básicamente un decodificador que consta sólo de lógica combinacional, que produce seis salidas a un display de 7 segmentos (el segmento *f* no se utiliza) para indicar el primer o el segundo piso, y dos salidas para las flechas luminosas indicadoras, que muestran la dirección en que se mueve el ascensor.

Entradas de la lógica de control de estados

- CLK es la señal de reloj que establece el tiempo que la puerta permanece abierta. El periodo del reloj es 10 segundos.
- *LLAMAR1* es la demanda al ascensor para ir al primer piso. El pulsador de demanda se encuentra en el exterior del ascensor, al lado de la puerta.
- *LLAMAR2* es la demanda al ascensor para ir al segundo piso. El pulsador de demanda se encuentra en el exterior del ascensor, al lado de la puerta.
- *PISO2* es la instrucción para que el ascensor vaya al segundo piso. Este pulsador se encuentra en el interior del ascensor.

- *PISO1* es la instrucción para que el ascensor vaya al primer piso. Este pulsador se encuentra en el interior del ascensor.
- *ABRIR* es la instrucción para que la puerta se abra cuando el ascensor está parado. Este pulsador se encuentra en el interior del ascensor.
- *LLEGADA* es la indicación de que el ascensor ha llegado a un nuevo piso. Un sensor en cada piso, activado por el ascensor, proporciona esta entrada.

Salidas de la lógica de control de estados

- *L1PB* y *L1PB_* son las salidas del latch del primer piso correspondientes a los pulsadores. Cuando se presiona uno de los pulsadores del primer piso, la solicitud se almacena inmediatamente en un latch S-R, independiente de la señal de reloj. *L1PB* y *L1PB_* son complementarias.
- *L2PB* y *L2PB_* son las salidas del latch del segundo piso correspondientes a los pulsadores. Cuando se presiona uno de los pulsadores del segundo piso, la solicitud se almacena inmediatamente en un latch S-R, independiente de la señal de reloj. *L2PB* y *L2PB_* son complementarias.
- *PUERTA* controla el estado de la puerta del ascensor. Cuando *PUERTA* = 0, la puerta está abierta. Cuando *PUERTA* = 1, la puerta está cerrada.
- *MOVIM* controla el movimiento del ascensor. Cuando *MOVIM* = 0, el ascensor está en espera (no se mueve). Cuando *MOVIM* = 1, el ascensor se mueve hacia arriba o hacia abajo.
- *DIR* controla la dirección del movimiento. Cuando *DIR* = 0, el ascensor sube para atender la siguiente llamada del segundo piso. Cuando *DIR* = 1, el ascensor baja para atender la siguiente llamada del primer piso.

Diagrama de estados de la lógica de control de estados del ascensor

Los seis estados del sistema son los siguientes:

- ESPERA1. El ascensor está en espera en el primer piso con la puerta abierta.
- CERRADO1. El ascensor está en espera en el primer piso con la puerta cerrada.
- SUBIDA. El ascensor sube del primer piso al segundo.
- ESPERA2. El ascensor está en espera en el segundo piso con la puerta abierta
- CERRADO2. El ascensor está en espera en el segundo piso con la puerta cerrada.
- BAJADA. El ascensor baja del segundo piso al primero.

El diagrama de estados de la Figura 11.15 muestra las condiciones del sistema para el paso de un estado al siguiente.

Descripción de funcionamiento

La descripción está referida al diagrama de estados de la Figura 11.15. Supongamos que el sistema está en el estado ESPERA1, el ascensor espera en el primer piso con la puerta abierta. Si después de un intervalo de tiempo establecido por el reloj, no se le llama desde el segundo piso y no se presiona el pulsador PISO2 ($\overline{LLAMAR2}$ x $\overline{PISO2}$), el sistema pasa al estado CERRADO1, en el que el ascensor permanece esperando en el primer piso con la puerta cerrada. Cuando el sistema está en el estado ESPERA1, y se le llama desde el segundo piso o se presiona el pulsador PISO2 (*LLAMAR2 + PISO2*), el sistema se salta el estado CERRADO1 y pasa al estado SUBIDA en el siguiente impulso de reloj. En el estado SUBIDA, la puerta se cierra y el ascensor sube.

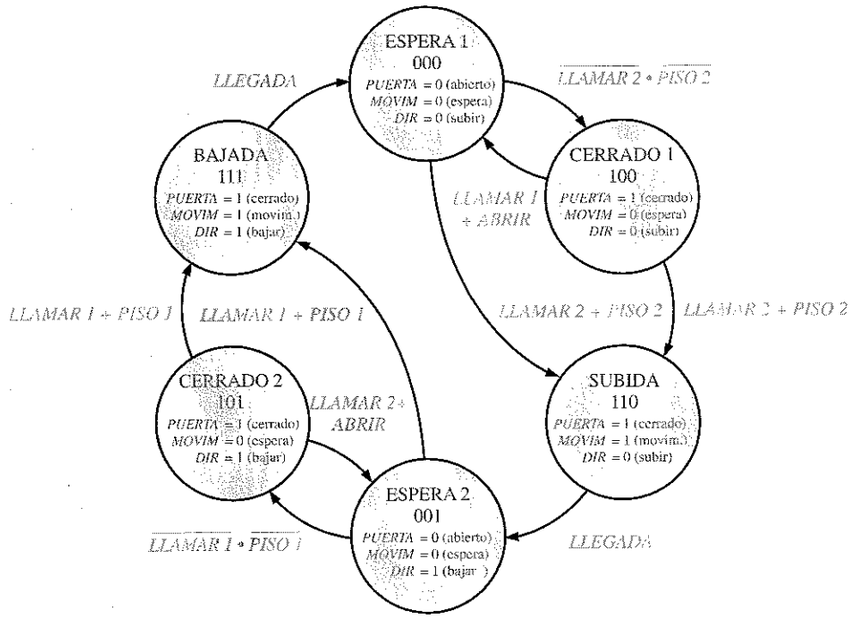


FIGURA 11.15
Diagrama de estados para el sistema de control del ascensor.

Cuando el sistema está en el estado CERRADO1, permanece en él hasta que se le llama desde el segundo piso o se presiona el botón PISO2 (*LLAMAR2 + PISO2*), lo que hace que el sistema pase al estado SUBIDA. Alternativamente, permanece en el estado CERRADO1 hasta que se le llama desde el mismo piso primero o se presiona el botón ABRIR (*LLAMAR1 + ABRIR*), lo que hace que el sistema vuelva al estado ESPERA1.

El sistema permanece en el estado de SUBIDA hasta que recibe la señal *LLEGADA*. Esta señal se produce cuando el ascensor llega al segundo piso y activa un conmutador que pone al sistema en estado ESPERA2. En este estado, el ascensor espera en el segundo piso con la puerta abierta. Si después de un periodo de tiempo establecido por el reloj, no se le llama desde el primer piso y no se presiona el botón PISO1 ($\overline{LLAMAR1}$ x $\overline{PISO1}$), el sistema pasa al estado CERRADO2, en el que el ascensor permanece esperando en el segundo piso con la puerta cerrada. Cuando el sistema está en el estado ESPERA2, y se le llama desde el primer piso o se pulsa el botón PISO1 (*LLAMAR1 + PISO1*), el sistema se salta el estado CERRADO2 y pasa

al estado BAJADA en el siguiente impulso de reloj. En el estado BAJADA, la puerta se cierra y el ascensor baja.

Cuando el sistema está en el estado CERRADO2, permanece en él hasta que se le llama desde el primer piso o se presiona el botón PISO1 (*LLAMAR1 + PISO1*), lo que hace que el sistema pase al estado BAJADA; o permanezca en el estado CERRADO2 hasta que se le llama desde el mismo piso segundo o se presiona el botón ABRIR (*LLAMAR2 + ABRIR*), lo que hace que el sistema vuelva al estado ESPERA2.

El sistema permanece en estado BAJAR hasta que se recibe la señal *LLEGADA* cuando el ascensor llega al primer piso, lo que activa un interruptor y pone al sistema en estado ESPERA1.

Implementación de la lógica de control con un PLD

Según el esquema de bloques de la GAL16V8 de la Figura 11.1(b), la Figura 11.16 presenta la asignación de pines para la GAL16V8 utilizada en la parte de la lógica de control de estados del sistema. Se va a desarrollar cada uno de los elementos que describen el sistema en lenguaje ABEL, y luego se unirán para formar el archivo de entrada completo. Obsérvese que la lógica de control tiene dos partes, los latches para las entradas de los pulsadores y la lógica de la secuencia de estados.

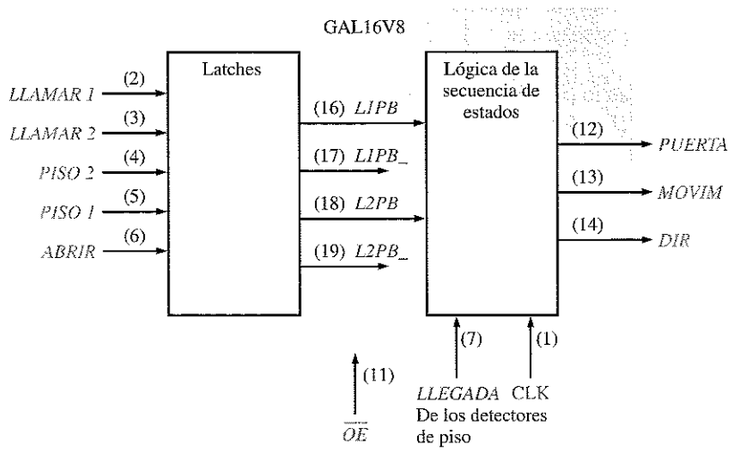


FIGURA 11.16
Diagrama de bloques de la lógica de control de estados con la asignación de pines de la GAL16V8 para las entradas y las salidas.

Entradas y salidas de la lógica de control de estados

A continuación se detallan las instrucciones que definen las entradas. $\overline{\text{OE}}$ es la activación de salida (Output Enable) que precisa la GAL16V8. La especificación de la numeración de pines es opcional.

```
CLK, !OE                          pin 1,11;
LLAMAR1, LLAMAR2                  pin 2,3;
PISO2, PISO1, ABRIR, LLEGADA      pin 4,5,6,7;
```


Puesto que el periodo de la señal de reloj es muy largo ($T = 10$ s), los latches de los pulsadores (PB) del primer piso y del segundo se implementan con las puertas de la matriz lógica para conseguir una respuesta inmediata, en lugar de utilizar los flip-flops D dependientes del reloj de las OLMC. Las instrucciones que definen las salidas se escriben como sigue:

```
L1PB, L1PB_              pin 16, 17 ISTYPE 'com,buffer;
L2PB, L2PB_              pin 18, 19 ISTYPE 'com,buffer;
PUERTA, MOVIM, DIR       pin 12, 13, 14 ISTYPE 'reg,buffer';
```

Las variables *L1PB, L1PB_, L2PB* y *L2PB_* producidas por los latches se enumeran sólo como salidas, porque se usan como entradas en la parte secuencial de la lógica de control de estados, y deben tener una asignación de pines para realimentarla a la lógica de la matriz. Estas salidas no se utilizan para propósitos externos.

Definición de estados

Los seis estados del sistema de control de estados deben definirse con sus valores binarios. Como ya hemos visto, el estado del sistema viene determinado por las variables de salida de control *PUERTA, MOVIM* y *DIR*. En primer lugar, estas variables de control de estados deben definirse en una instrucción tal como

```
ESTADCON    = [PUERTA, MOVIM, DIR];
```

A continuación, se deben establecer los valores binarios, para lo que hay dos métodos. El primero utiliza la notación de conjuntos. Por ejemplo, el estado ESPERA1 sería:

```
ESPERA1 = [0 , 0 , 0];
```

El segundo método utiliza la notación binaria en ABEL, en la que el número binario va precedido de ^B del siguiente modo:

```
ESPERA1     = ^B000;
CERRADO1    = ^B100;
SUBIDA      = ^B110;
ESPERA2     = ^B001;
CERRADO2    = ^B101;
BAJADA      = ^B111;
```

La lógica de los latches de los pulsadores

Los dos latches para las entradas de los pulsadores se implementan con lógica combinacional. Es necesario hacerlo así porque los pulsadores se pueden presionar y liberar en cualquier instante y, por tanto, los flip-flops sincronizados con la señal de reloj de las OLMC no se pueden utilizar. Por ejemplo, se puede presionar y liberar un pulsador entre impulsos de reloj (su periodo es de 10 segundos) y ser completamente ignorado si se usa un flip-flop para almacenar el cierre del interruptor. Sin embargo, un latch S-R almacena inmediatamente el cierre del interruptor y lo mantiene hasta que se desactiva.

El latch para el primer piso está en estado SET (*L1PB* = 1) si se presiona el pulsador de llamada del primer piso *(LLAMADA1)* o el ascensor está en el primer piso y se presiona el pulsador ABRIR, $(\overline{DIR})$ *(ABRIR)*, o si el ascensor está en el segundo piso y se presiona el pulsador PISO1, *(DIR) (PISO1)*.

El ascensor está en el primer piso cuando el sistema está en los estados ESPERA1 o CERRADO1. El ascensor está en el segundo piso cuando el sistema está en los estados ESPERA2 o CERRADO2. En el diagrama de tiempos puede verse que la variable *DIR* define el piso, ya que *DIR* = 0 cuando el sistema está en el estado ESPERA1 o CERRADO1, y *DIR* = 1 cuando el sistema está en ESPERA2 o CERRADO2. La condición del estado SET para el latch del primer piso viene dada por la siguiente expresión booleana:

$$L1PB = LLAMADA1 + (\overline{DIR})\,(ABRIR) + (DIR)\,(PISO1) + \overline{L1PB\_}$$

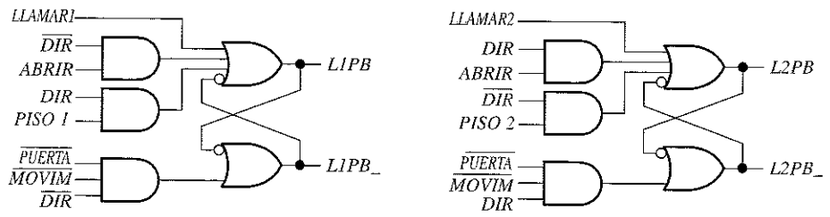
El latch está en estado RESET si el sistema está en el estado ESPERA1 (000). La condición del estado RESET viene dada por la siguiente expresión booleana

$$L1PB\_ = (\overline{PUERTA})\,(\overline{MOVIM})\,(\overline{DIR}) + \overline{L1PB}$$

Las ecuaciones para el latch del segundo piso son similares

$$L2PB = LLAMADA2 + (DIR)\,(ABRIR) + (\overline{DIR})\,(PISO2) + \overline{L2PB\_}$$
$$L2PB\_ = (\overline{PUERTA})\,(\overline{MOVIM})\,(DIR) + \overline{L2PB}$$

En la Figura 11.17 se muestra la lógica combinacional para los latches del primer y segundo pisos.



(a) Latch para el pulsador del primer piso

(b) Latch para el pulsador del segundo piso

FIGURA 11.17
Circuitos latches de la lógica de control de estados.

Las ecuaciones booleanas para los latches expresadas en ABEL son:

```
L1PB = LLAMAR1 # !DIR.FB & ABRIR # DIR.FB & PISO1 # !L1PB_;
L1PB_ = (!PUERTA.FB & !MOVIM.FB & !DIR.FB) # !L1PB;
L2PB = LLAMAR2 # DIR.FB & ABRIR # !DIR.FB & PISO2 # !L2PB_;
L2PB_ = (!PUERTA.FB & !MOVIM.FB & (DIR.FB) # !L2PB;
```

El uso de la extensión .FB con las variables de estado indica que la variable se realimenta directamente a la lógica de la matriz a partir del flip-flop asociado. Los flip-flops de la OLMC producen las variables de estado *PUERTA, MOVIM* y *DIR* en la etapa secuencial del sistema.

La lógica de la secuencia de estados

La parte secuencial de la lógica de control de estados genera las salidas de las variables de estado *PUERTA, MOVIM* y *DIR*. Esta lógica se puede definir utilizando el método de entrada del diagrama de estados, no siendo necesario desarrollar las ecuaciones para la lógica combinacional del latch. Las instrucciones IF-THEN-ELSE para cada uno de los seis estados son las siguientes. Las variables de entrada/salida se encierran entre paréntesis.

```
State ESPERA1:  if (L2PB) then SUBIR else CERRADO1;
State CERRADO1: if (L2PB) then SUBIR else if L1PB then ESPERA1 else CERRADO1;
State SUBIDA:   if (LLEGADA) then ESPERA2 else SUBIR;
State ESPERA2:  if (L1PB) then BAJAR else CERRADO2;
State CERRADO2: if (L1PB) then BAJAR else if L2PB then ESPERA2 else CERRADO2;
State BAJADA:   if (LLEGADA) then ESPERA1 else BAJAR;
```

Como puede verse, un sistema secuencial puede implementarse en un PLD sin tener que diseñar primero la lógica, ya que el software implementa automáticamente el diseño basándose en una descripción del diagrama de estados.

EJEMPLO 11.6 Escribir el archivo de entrada ABEL completo para la lógica del sistema de control, utilizando los elementos ya desarrollados.

Solución

```
Module  Control_Estados_Ascensor
Title   'Lógica de control para el sistema del ascensor para dos pisos'
"Declaración de dispositivo
     Contr_ascensor        device         'P16V8'
"Declaración de pines
     CLK, !OE                          pin 1,11;
     LLAMAR1, LLAMAR2                  pin 2,3;
     PISO2, PISO1, ABRIR, LLEGADA      pin 4,5,6,7;
     L1PB, L1PB_                pin 16, 17 ISTYPE 'com,buffer;
     L2PB, P2PB_                pin 18, 19 ISTYPE 'com,buffer;
     PUERTA, MOVIM, DIR         pin 12, 13, 14 ISTYPE 'reg,buffer';
"Definición de estados
     ESTADCON = [PUERTA, MOVIM, DIR];
     ESPERA1  = ^B000;
     CERRADO1 = ^B100;
     SUBIDA   = ^B110;
     ESPERA2  = ^B001;
     CERRADO2 = ^B101;
     BAJADA   = ^B111;
Equations
     ESTADCON.CLK = CLK;
     L1PB = LLAMAR1 # !DIR.FB & ABRIR # DIR.FB & BAJAR # !L1PB_;
```

```
    L1PB_ = (!PUERTA.FB & !MOVIM.FB & !DIR.FB) # !L1PB;
    L2PB = LLAMAR2 # DIR.FB & ABRIR # !DIR.FB & SUBIR # !L2PB_;
    L2PB_ = (!PUERTA.FB & !MOVIM.FB & (DIR.FB) # !L2PB;

State_diagram ESTADCON

State ESPERA1:   if (L2PB) then SUBIR else CERRADO1;
State CERRADO1:  if (L2PB) then SUBIR else if L1PB then ESPERA1 else
                 CERRADO1;
State SUBIDA:    if (LLEGADA) then ESPERA2 else PISO2;
State ESPERA2:   if (L1PB) then BAJAR else CERRADO2;
State CERRADO2:  if (L1PB) then BAJAR else if L2PB then ESPERA2 else
                 CERRADO2;
State BAJADA:    if (LLEGADA) then ESPERA1 else PISO1;

Test_Vectors ([CLK, LLAMAR1, LLAMAR2, PISO2, PISO1, ABRIR, LLEGADA] -> [PUERTA, MOVIM, DIR])
              [.c.,  0  ,  0  ,  0 ,  0 ,  0 ,   1  ] ->[  .x.,  .x.,.x.];
              [.c.,  1  ,  0  ,  0 ,  0 ,  0 ,   1  ] ->[  .x.,  .x.,.x.];
              [.c.,  1  ,  0  ,  0 ,  0 ,  0 ,   1  ] ->[   0 ,   0 , 0 ];
              [.c.,  0  ,  0  ,  0 ,  0 ,  0 ,   0  ] ->[   1 ,   0 , 0 ];
              [.c.,  1  ,  0  ,  0 ,  0 ,  0 ,   0  ] ->[   0 ,   0 , 0 ];
              [.c.,  0  ,  0  ,  0 ,  0 ,  0 ,   1  ] ->[   1 ,   0 , 0 ];
              [.c.,  0  ,  0  ,  0 ,  0 ,  1 ,   0  ] ->[   0 ,   0 , 0 ];
              [.c.,  0  ,  1  ,  0 ,  0 ,  0 ,   0  ] ->[   1 ,   1 , 0 ];
              [.c.,  0  ,  0  ,  0 ,  0 ,  0 ,   1  ] ->[   0 ,   0 , 1 ];
              [.c.,  0  ,  0  ,  0 ,  0 ,  0 ,   0  ] ->[   1 ,   0 , 1 ];
              [.c.,  0  ,  1  ,  0 ,  0 ,  0 ,   0  ] ->[   0 ,   0 , 1 ];
              [.c.,  0  ,  0  ,  0 ,  0 ,  0 ,   0  ] ->[   1 ,   0 , 1 ];
              [.c.,  0  ,  0  ,  0 ,  0 ,  1 ,   0  ] ->[   0 ,   0 , 1 ];
              [.c.,  1  ,  0  ,  0 ,  0 ,  0 ,   0  ] ->[   1 ,   1 , 1 ];
              [.c.,  0  ,  0  ,  1 ,  0 ,  0 ,   0  ] ->[   0 ,   0 , 1 ];
              [.c.,  0  ,  0  ,  0 ,  0 ,  0 ,   1  ] ->[   1 ,   1 , 0 ];
              [.c.,  0  ,  0  ,  0 ,  1 ,  0 ,   0  ] ->[   1 ,   1 , 1 ];
              [.c.,  0  ,  0  ,  0 ,  0 ,  0 ,   1  ] ->[   0 ,   0 , 0 ];
              [.c.,  0  ,  0  ,  0 ,  0 ,  0 ,   0  ] ->[   1 ,   0 , 0 ];
              [.c.,  0  ,  1  ,  0 ,  0 ,  0 ,   0  ] ->[   1 ,   1 , 0 ];
              [.c.,  0  ,  0  ,  0 ,  0 ,  0 ,   1  ] ->[   0 ,   0 , 1 ];
              [.c.,  0  ,  0  ,  0 ,  0 ,  0 ,   0  ] ->[   1 ,   0 , 1 ];
              [.c.,  0  ,  0  ,  0 ,  1 ,  0 ,   0  ] ->[   1 ,   1 , 1 ];
              [.c.,  0  ,  0  ,  0 ,  0 ,  0 ,   1  ] ->[   0 ,   0 , 0 ];

END
```

Problema relacionado. Los vectores de prueba deberían representar todos los posibles caminos del diagrama de estados. Verificar que es así.

Implementación de la lógica del display con un PLD

En la Figura 11.18 se muestra la asignación de pines para la GAL16V8 utilizada en la parte de la lógica del display del sistema. Las entradas a la lógica del display proceden de la lógica de control de estados. Sólo se requieren seis salidas activas a nivel

BAJO para presentar la lectura en el display de 7 segmentos, ya que sólo se emplean el dígito 1 y el 2; por tanto, el segmento *f* no es necesario. Otras dos salidas activas a nivel BAJO sirven para excitar las flechas indicadoras hacia arriba y hacia abajo. Toda la lógica de esta parte del sistema es tipo combinacional.

Declaración de los pines de entrada y salida

La instrucción ABEL que designa los pines de entrada se puede escribir como sigue. $\overline{OE}$ es la salida de activación, que requiere la GAL16V8.

```
!OE, MOVIM, DIR        pin 11,2,3
```

Las instrucciones ABEL que designan los pines de salida se pueden escribir de la siguiente forma:

```
SEGA, SEGB, SEGC       pin 12, 13, 14      ISTYPE 'com,invert';
SEGD, SEGE, SEGG       pin 15, 16, 17      ISTYPE 'com,invert';
FLSUB, FLBAJ           pin 18, 19          ISTYPE 'com,invert';
```

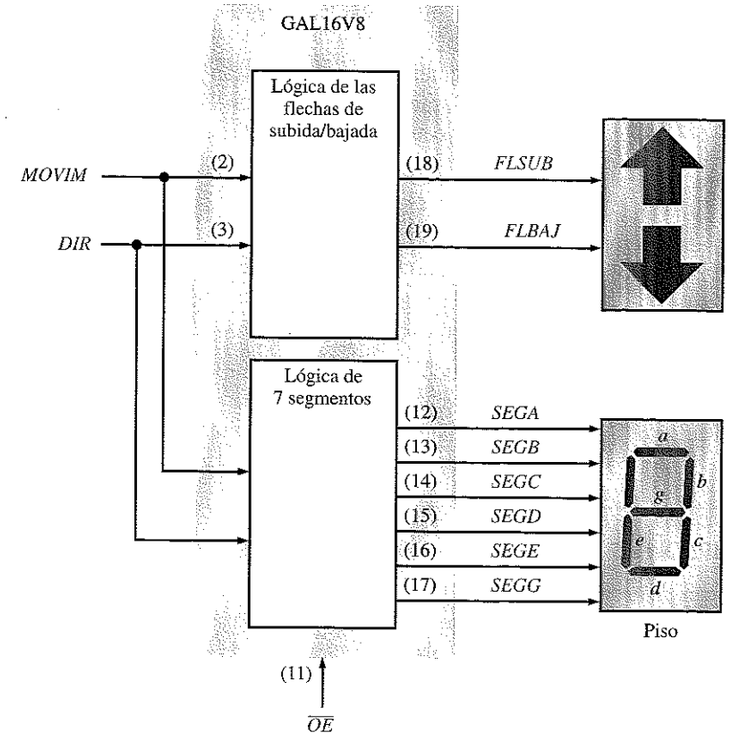


FIGURA 11.18
Diagrama de bloques de la lógica del display, que muestra la asignación de pines para las entradas y las salidas.

La lógica de la flechas indicadoras de subida y bajada

La flecha de subida debe iluminarse cuando el ascensor está en movimiento (*MOVIM* = 1) hacia arriba (*DIR* = 0). La ecuación booleana es:

$$FLSUB = MOVIM \cdot \overline{DIR} \cdot \overline{OE}$$

La flecha de bajada debe iluminarse cuando el ascensor está en movimiento (*MOVIM* = 1) hacia abajo (*DIR* = 1). La ecuación booleana es:

$$FLBAJ = MOVIM \cdot DIR \cdot \overline{OE}$$

Como puede verse, estas funciones se pueden implementar con puertas AND de 2 entradas.

Las ecuaciones booleanas para los latches en ABEL son:

```
FLSUB = MOVIM & !DIR &!OE;
FLBAJ = MOVIM & DIR & !OE;
```

La lógica de 7 segmentos

El display debe presentar un "1" cuando el sistema está en los estados ESPERA1 y CERRADO1. El display debe presentar un "2" cuando está en los estados ESPERA2 y CERRADO2.

Los segmentos *b* y *c* se emplean para el "1", y los segmentos *a, b, d, e* y *g* se usan para el "2", como muestra la Figura 11.18. El segmento *b* es común a los dos dígitos. Por tanto, las ecuaciones en ABEL de los segmentos son las siguientes:

```
SEGA = !MOVIM & DIR & !OE;
SEGB = !MOVIM & !OE;
SEGC = !MOVIM & !DIR & !OE;
SEGD = !MOVIM & DIR & !OE;
SEGE = !MOVIM & DIR & !OE;
SEGG = !MOVIM & DIR & !OE;
```

Obsérvese que las ecuaciones para los segmentos *a, d, e* y *g* son idénticas, luego se podría utilizar una única salida para todas ellas. Sin embargo, esta salida se tendría que separar externamente para excitar a los cuatro segmentos.

EJEMPLO 11.7 Escribir un archivo de entrada ABEL para la lógica del display del sistema utilizando los elementos ya desarrollados.

Solución

```
Module Lógica_Display_Ascensor
Title 'Lógica del display para el sistema de ascensor de dos pisos'
"Declaración de dispositivos
     Display_ascn      device      'P16V8'
"Declaración de pines
     !OE, MOVIM, DIR        pin 11,2,3
     SEGA, SEGB, SEGC       pin 12, 13, 14   ISTYPE 'com,invert';
     SEGD, SEGE, SEGG       pin 15, 16, 17   ISTYPE 'com,invert';
     FLSUB,FLBAJ            pin 18, 19       ISTYPE 'com,invert';
Equations
     FLSUB  = MOVIM & !DIR & !OE;
     FLBAJ  = MOVIM & DIR & !OE;
     SEGA   = !MOVIM & DIR & !OE;
     SEGB   = !MOVIM & !OE;
     SEGC   = !MOVIM & !DIR & !OE;
     SEGD   = !MOVIM & DIR & !OE;
     SEGE   = !MOVIM & DIR & !OE;
     SEGG   = !MOVIM & DIR & !OE;
Test_Vectors
([!OE, MOVIM, DIR] -> [SEGA, SEGB, SEGC, SEGD, SEGE, SEGG, FLSUB, FLBAJ])
 [ 0 ,  0  ,  0 ] -> [ 1  ,  0 ,  0 ,  1 ,  1 ,  1 ,  1  ,  1 ];
 [ 0 ,  0  ,  1 ] -> [ 0  ,  0 ,  1 ,  0 ,  0 ,  0 ,  1  ,  1 ];
 [ 0 ,  1  ,  0 ] -> [ 1  ,  1 ,  1 ,  1 ,  1 ,  1 ,  0  ,  1 ];
 [ 0 ,  1  ,  1 ] -> [ 1  ,  1 ,  1 ,  1 ,  1 ,  1 ,  1  ,  0 ];
 [ 1 , .x. , .x.] -> [ 1  ,  1 ,  1 ,  1 ,  1 ,  1 ,  1  ,  1 ];
END
```

Problema relacionado. Rehacer el archivo de entrada para combinar las cuatro salidas de segmentos que son idénticas, en una salida denominada SEGADEG.

REVISIÓN DE LA SECCIÓN 11.5

1. Explicar por qué se requieren latches en la lógica de control del sistema.
2. En los vectores de prueba del Ejemplo 11.6, explicar cómo se inicializa el sistema en el estado ESPERA1. ¿Por qué son las variables de salida de las dos primeras filas "indiferentes"?
3. ¿Se puede utilizar el método de entrada del diagrama de estados en lugar de las ecuaciones para la lógica del display? Explicarlo.
4. ¿Se puede utilizar el método de entrada de la tabla de verdad para el diseño lógico tratado en esta sección?

11.6 ■ APLICACIÓN A LOS SISTEMAS DIGITALES

En el Capítulo 9 se implementó la lógica secuencial del sistema de control de semáforos utilizando dispositivos SSI y MSI. Ahora, se van a aplicar los nuevos conocimientos sobre dispositivos PLD y sobre su programación, para implementar la lógica secuencial con una GAL22V10. Es aconsejable que revise la Sección 9.10 antes de continuar.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Programar la GAL22V10 a partir del diagrama de estados. ■ Explicar cómo reemplaza la GAL a la lógica implementada en el Capítulo 9. ■ Desarrollar un archivo de entrada ABEL para implementar el sistema de control de semáforos.

Diagrama de bloques de la lógica secuencial

En la Figura 11.19 se presenta el esquema de bloques del sistema de control de semáforos y en la Figura 11.20 el diagrama de bloques de la lógica secuencial.

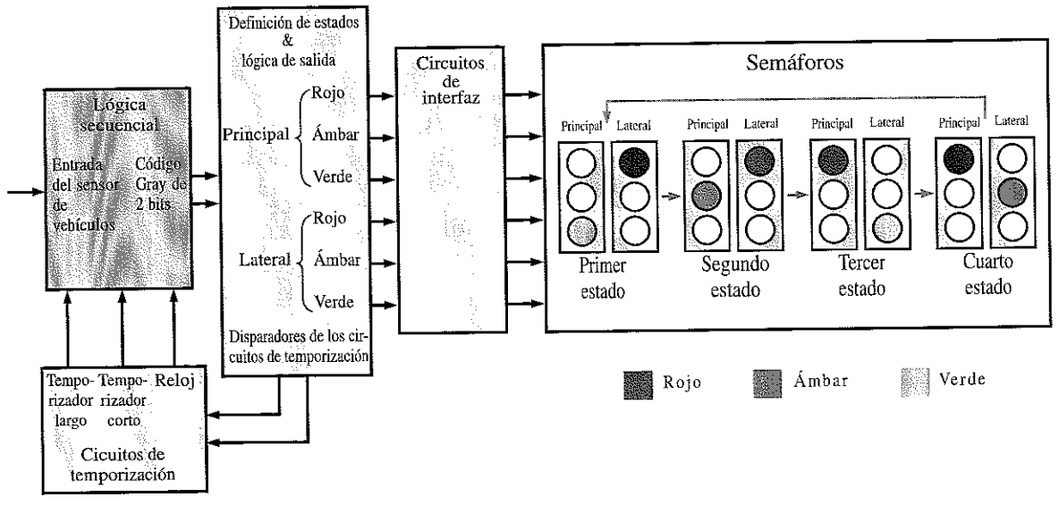


FIGURA 11.19
Diagrama de bloques del sistema de control de semáforos.

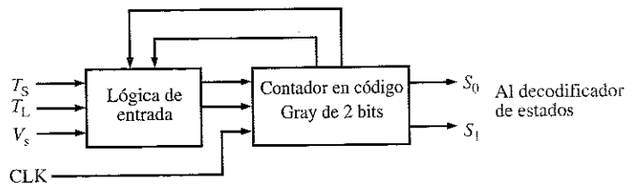
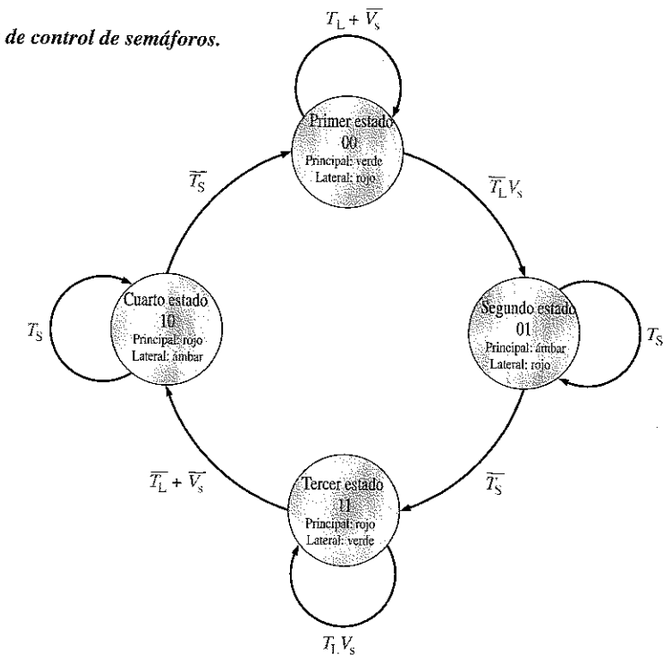


FIGURA 11.20
Diagrama de bloques de la lógica secuencial.

La lógica secuencial del sistema utiliza un contador en código Gray de 2 bits para generar la secuencia de estados descrita en el diagrama de estados de la Figura 11.21. A continuación, repasamos las variables de entrada:

V_S	Vehículo en la calle lateral
$\overline{V}_S$	No hay vehículo en la calle lateral
T_L	Temporizador largo activado (25 s)
$\overline{T}_L$	Temporizador largo desactivado (25 s)
T_S	Temporizador corto activado (4 s)
$\overline{T}_S$	Temporizador corto desactivado (25 s)

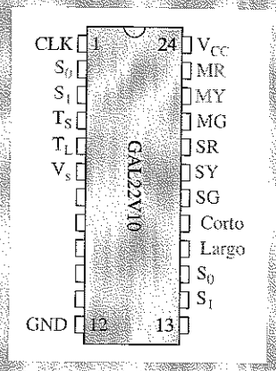
FIGURA 11.21
Diagrama de estados del sistema de control de semáforos.



■ PRÁCTICA DE SISTEMAS 1: Programación de la GAL22V10

La GAL22V10 que se ha utilizado en la sección de aplicaciones a los sistemas digitales del Capítulo 7 para la decodificación de estados y la lógica de salida, se puede programar para añadir la lógica secuencial, ya que todavía quedan bastantes entradas y salidas disponibles.

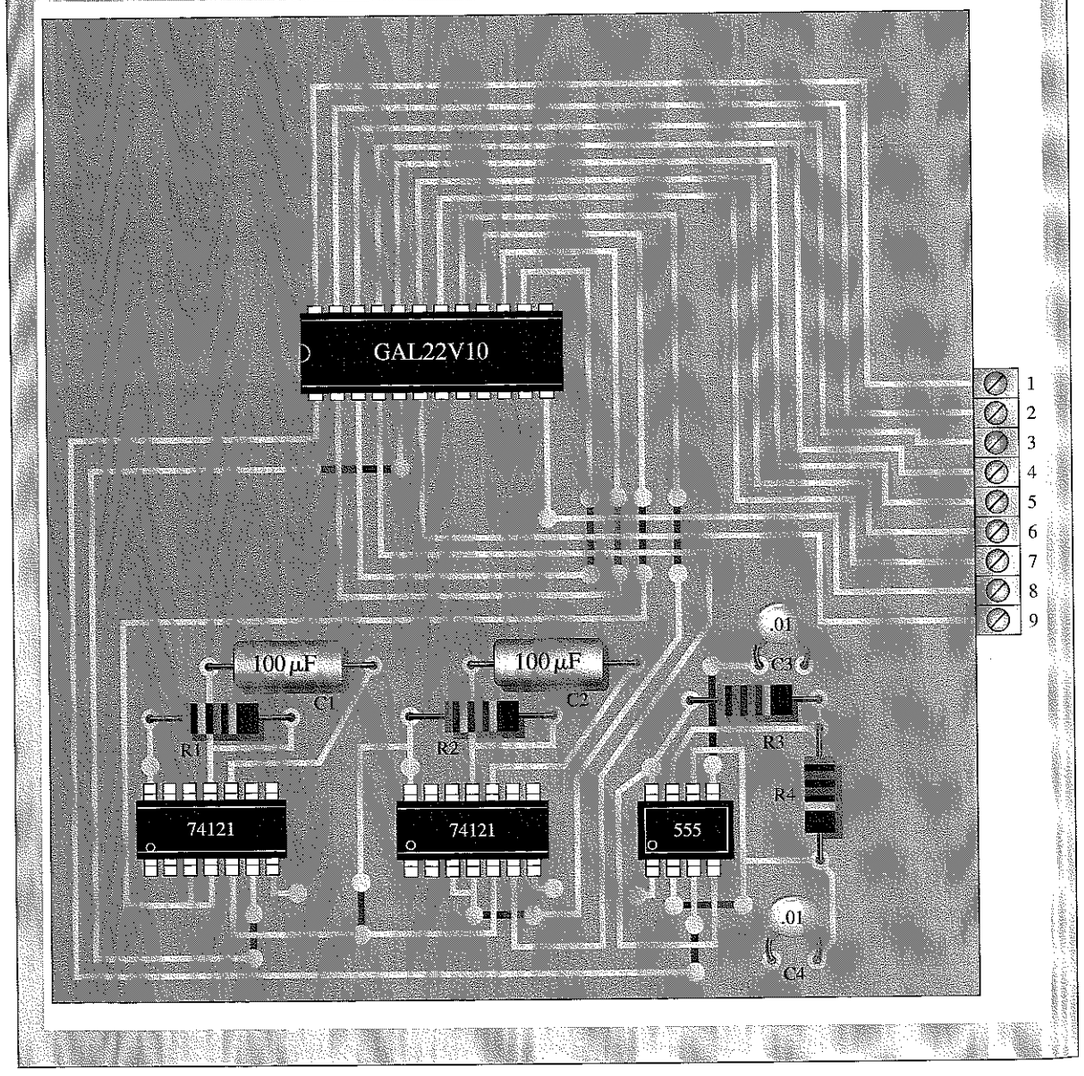
- ***Actividad 1.*** A partir del diagrama de estados de la Figura 11.21 y del diagrama de asignación de pines de la GAL22V10 que se muestra en esta sección, escribir un archivo de entrada en ABEL para programar la lógica secuencial en el dispositivo. Las entradas y salidas que se han programado anteriormente para la decodificación de estados y la lógica de salida se indican en gris. Las entradas y salidas correspondientes a la lógica secuencial se muestran en negro.
- ***Actividad 2.*** Escribir un sencillo archivo de entrada para el sistema completo, en el que se combinen adecuadamente el archivo de entrada para la decodificación de estados y la lógica de salida del Capítulo 7, y el archivo de entrada para la lógica secuencial que debe elaborar. Los vectores de prueba para este archivo de entrada completo se desarrollan en la Práctica de sistemas 2.



■ PRÁCTICA DE SISTEMAS 2: Verificación y pruebas

Una única GAL22V10 reemplaza a los dispositivos 74LS74A, 74LS08 y 74LS153 que se usaron originalmente en el Capítulo 9 para implementar la lógica secuencial, así como el 74LS139 y el 74LS08 utilizados en el Capítulo 6 para la decodificación de estados y la lógica de salida. Los circuitos de temporización son los mismos.

- ***Actividad 1.*** Desarrollar los vectores de prueba para la GAL programada, y añadir el archivo de entrada de la Práctica de sistemas 1.
- ***Actividad 2.*** Identificar la función de cada terminal de entrada y de salida en la tarjeta de circuito impreso que se muestra más abajo.
- ***Actividad 3.*** Dibujar un diagrama lógico de la tarjeta del sistema.



REVISIÓN DE LA SECCIÓN 11.6

1. Después de haber programado en esta sección la GAL22V10, ¿cuántas salidas quedan todavía disponibles que no se usan?
2. ¿Cuántas entradas hay disponibles que no se utilizan?
3. Comparar la implementación de toda la lógica cuando se usa la GAL y cuando se usan dispositivos SSI y MSI.

■ RESUMEN

- La GAL16V8 y la GAL22V10 son ejemplos de dispositivos lógicos reprogramables.
- La OLMC de una GAL es la macrocelda lógica de salida que se puede configurar por programación para obtener cualquier salida combinacional o secuencial.
- En el modo secuencial de la OLMC, la salida se hace a través de un flip-flop D.
- La instrucción ISTYPE del lenguaje ABEL se puede utilizar para declarar una salida como combinacional o secuencial, mediante ISTYPE 'com'; o ISTYPE 'reg';.
- Los operadores de asignación de ABEL : = y :> se usan para indicar una salida secuencial.
- La extensión .CLK se usa para indicar que el dispositivo de registro es un flip-flop sincronizado con la señal de reloj.
- La extensión .AR se usa para poner a cero de forma asíncrona un flip-flop.
- Los dos tipos básicos de máquinas de estados son Moore y Mealy.
- En una máquina de estados tipo Moore, las salidas se determinan sólo en función del estado interno y de las entradas síncronas.
- En una máquina de estados tipo Mealy, las salidas se determinan sólo en función del estado interno y de las entradas no síncronas del sistema.
- Un contador es un ejemplo básico de una máquina de estados tipo Moore.
- Las máquinas de estados se pueden implementar en un PLD utilizando el método de entrada del diagrama de estados en lenguaje ABEL. La instrucción IF-THEN-ELSE es un elemento importante de la sintaxis del diagrama de estados.

■ PALABRAS CLAVE

Estos términos también se encuentran en el glosario del final del libro.

Celda global Celda programable de una matriz PLD que afecta a todas las OLMC cuando se programan.

Celda local Celda programable de una matriz PLD que afecta a las OLMC individuales cuando se programan.

Secuencial Configuración de salida de un PLD en que la salida procede de un flip-flop.

■ AUTOTEST

1. La GAL22V10 tiene
   (a) 10 entradas y 22 salidas **(b)** 22 entradas y 10 salidas
   (c) 11 entradas y 10 entradas/salidas **(d)** 11 entradas y 10 salidas
2. La GAL16V8 tiene
   (a) 8 entradas y 16 salidas **(b)** 16 entradas y 8 salidas
   (c) 8 entradas y 8 entradas/salidas **(d)** 8 entradas y 8 salidas

3. Una OLMC es
   (a) Celda principal del latch de salida
   (b) Celda de multiplexor lógica impar
   (c) Método de minimización lógico óptimo
   (d) Macrocelda lógica de salida
4. Un OLMC típica está formada por
   (a) puertas, multiplexores y un flip-flop
   (b) puertas y un registro de desplazamiento
   (c) un contador en código Gray
   (d) una matriz lógica fija
5. ¿Cuál de las siguientes ecuaciones ABEL especifica una salida secuencial?
   (a) X = A & B; (b) !X = !(A & B);
   (c) X := A & B; (d) X --> A & B;
6. Un contador se puede implementar en un PLD utilizando lenguaje ABEL mediante
   (a) introducción de las ecuaciones
   (b) introducción de la tabla de verdad
   (c) introducción del diagrama de estados
   (d) todas las respuestas anteriores
   (e) las respuestas (b) y (c)
7. La lógica combinacional puede implementarse en un PLD utilizando en lenguaje ABEL mediante
   (a) introducción de las ecuaciones
   (b) introducción de la tabla de verdad
   (c) introducción del diagrama de estados
   (d) todas las respuestas anteriores
   (e) las respuestas (a) y (b)
8. Una instrucción IF-THEN-ELSE válida es
   (a) if A then C else D
   (b) if A then C else D;
   (c) IF A THEN C ELSE D
   (d) IF-A-THEN-C-ELSE-D:
   (e) las respuestas (b) y (c)
9. La expresión ABEL A = [0, 1, 0]
   (a) significa que el estado A queda definido por el valor binario 010.
   (b) significa que la variable A tiene un secuencia de valores 0, 1, 0.
   (c) debe ir precedida de la instrucción de definición de estados
   (d) las respuestas (a) y (c)
10. .CLK y .AR son ejemplos de
    (a) constantes especiales
    (b) variables ABEL
    (c) extensiones
    (d) ninguna de las anteriores respuestas.

■ PROBLEMAS SECCIÓN 11.1 La OLMC completa

1. Enumerar las cuatro configuraciones de la GAL22V10.
2. Determinar el modo de configuración de la GAL16V8 para cada una de las combinaciones de valores de celdas siguientes:

   (a) $\overline{SYN} = 0, AC0 = 1, XOR = 1$
   (b) $\overline{SYN} = 1, AC0 = 0, XOR = 0$
3. Además del flip-flop y las puertas lógicas, ¿qué tipo de dispositivo se usa en una OLMC?

SECCIÓN 11.2 Selección de modo de las OLMC

4. Determinar el modo de la GAL22V10 para cada una de las combinaciones de bits siguientes.
   (a) $S_1S_0 = 10$ (b) $S_1S_0 = 00$
   (c) $S_1S_0 = 11$ (d) $S_1S_0 = 01$
5. Explicar cada una de las declaraciones de pines de salida en ABEL siguientes:
   (a) X pin 15 ISTYPE 'com';
   (b) Q2 pin 20 ISTYPE 'reg';
   (c) A PIN 18 ISTYPE 'com,invert';
   (d) Q1 pin 21 ISTYPE 'reg,buffer';
6. Identificar cada una de las expresiones ABEL como combinacional o secuencial
   (a) Y = A # B;
   (b) X := A & !B;
   (b) Q1 := D1 # D0;
   (d) Q2 = !C;
7. Escribir en ABEL una ecuación de reloj para acompañar a la siguiente expresión: Q1 : = (A & B) # C;

SECCIÓN 11.3 Implementación de registros de desplazamiento con dispositivos PLD

8. Escribir un archivo de entrada en ABEL, excluyendo los vectores de prueba, para implementar el registro de desplazamiento con entrada serie y salida paralelo de la Figura 11.22 en una GAL22V10.

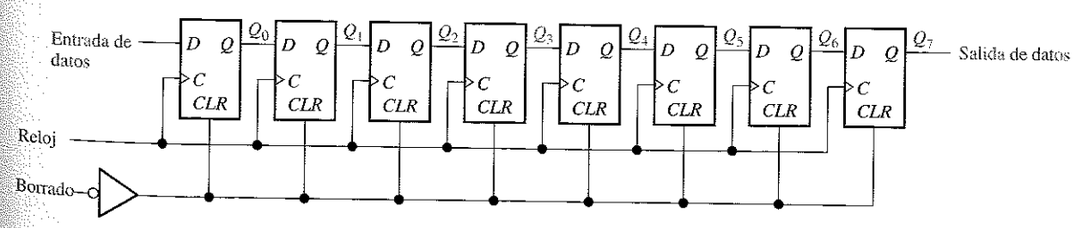
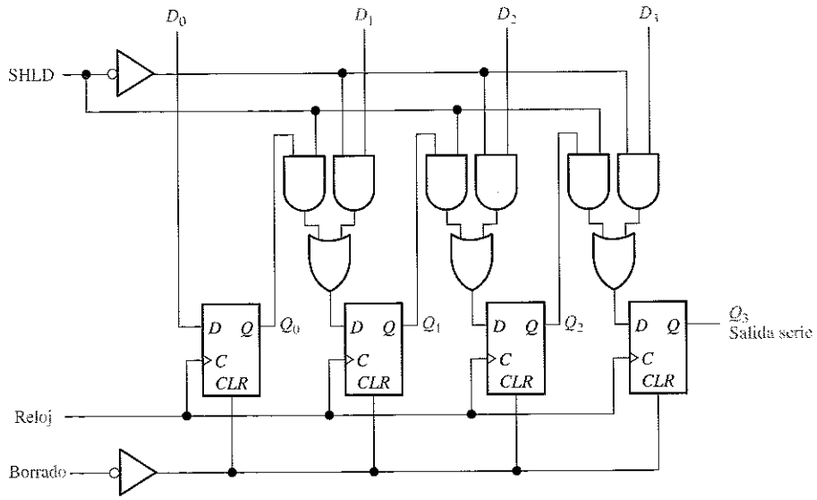


FIGURA 11.22

9. Desarrollar los vectores de prueba para el archivo de entrada del problema 8.
10. Repetir el problema 8 utilizando como dispositivo objeto una GAL16V8.
11. Repetir el problema 9 utilizando como dispositivo objeto una GAL16V8.
12. Escribir un archivo de entrada en ABEL para ampliar el registro de desplazamiento de 4 bits con entrada paralelo y salida serie de la Figura 11.23 a un registro de 8 bits. El dispositivo objeto es una GAL22V10. Incluir los vectores de prueba.

FIGURA 11.23



13. Determinar la posibilidad de implementar el registro de 8 bits del problema 12 con una GAL16V8.

SECCIÓN 11.4 Implementación de contadores con dispositivos PLD

14. Explicar la diferencia básica entre una máquina de estados Moore y una Mealy.

15. ¿Cuál es el mayor contador que se puede implementar con una única GAL16V8, en términos de número de etapas o bits?

16. ¿Cuál es el mayor contador que se puede implementar con una única GAL22V10, en términos de número de etapas o bits?

17. Convertir el contador en código Gray ascendente/descendente de 3 bits de la Figura 11.24 en un contador en código Gray ascendente de 3 bits, e implementarlo en una GAL22V10 utilizando como método de introducción las ecuaciones lógicas.

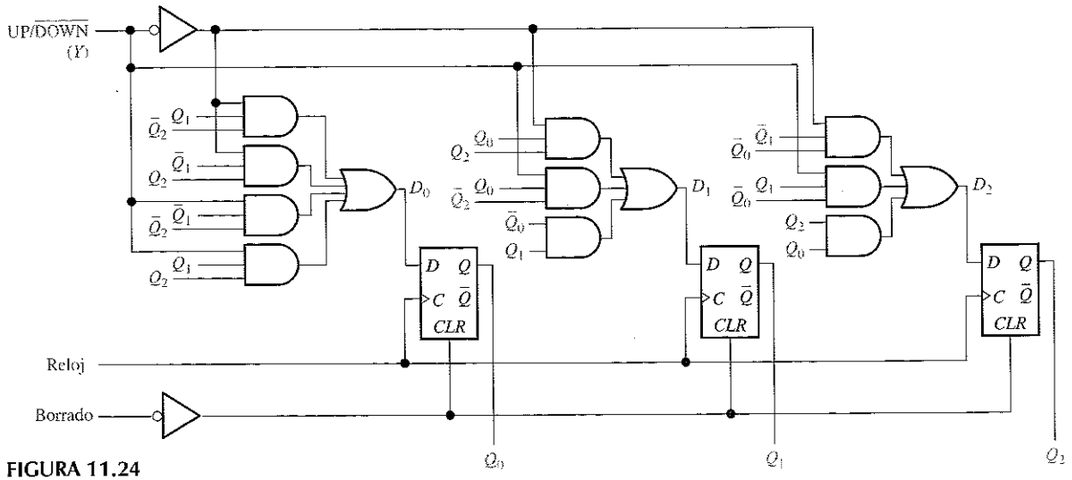


FIGURA 11.24

18. Repetir el problema 17 utilizando como método de introducción la tabla de verdad.

19. Repetir el problema 17 utilizando como método de introducción el diagrama de estados.

20. Repetir el problema 17 para una GAL16V8.

21. Repetir el problema 18 para una GAL16V8.

22. Repetir el problema 19 para una GAL16V8.

SECCIÓN 11.5 Implementación de sistemas mediante dispositivos PLD

23. Tomando como referencia el diagrama de estados de la Figura 11.25, si se presiona el pulsador ABRIR, decir qué ocurre cuando el sistema está en el estado

(a) CERRADO1
(b) ESPERA1
(c) SUBIDA
(d) ESPERA2
(e) CERRADO2
(f) BAJADA

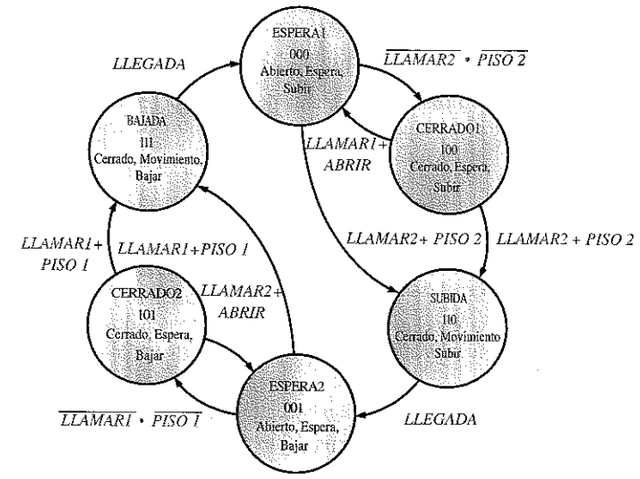


FIGURA 11.25

24. Repetir el problema 23 para cada uno de los siguientes casos:

(a) Se presiona el pulsador de PISO1
(b) Se presiona el pulsador de PISO2
(c) Se presiona el pulsador de LLAMAR1
(d) Se presiona el pulsador de LLAMAR2

25. Escribir un archivo de entrada completo en ABEL para implementar la lógica de control de estados del sistema del ascensor en una GAL22V10, en lugar de en una GAL16V8.

26. Repetir el problema 25 para la lógica del display del sistema de control del ascensor.

Aplicación a sistemas

SECCIÓN 11.6 Aplicación a los sistemas digitales

27. ¿Puede implementarse el sistema de control de semáforos con una única GAL16V8? Si no es posible, explicar por qué.

28. Suponer que la entrada del temporizador corto (T_S) de la lógica secuencial del sistema de control de semáforos falla, de modo que siempre hay un nivel ALTO ($T_S = 1$). ¿Cómo afecta al funcionamiento del sistema?

29. Suponer que la entrada del temporizador largo (T_L) de la lógica secuencial del sistema de control de semáforos falla, de modo que siempre hay un nivel BAJO ($T_L = 0$). ¿Cómo afecta al funcionamiento del sistema?
30. Si el sensor de vehículos siempre genera un nivel ALTO, ¿cómo afecta esto al funcionamiento del sistema?

Problemas especiales de diseño

31. Diseñar un registro de desplazamiento con entrada y salida serie de 8 bits, y escribir el archivo de entrada en ABEL para implementarlo utilizando dispositivos GAL16V8.
32. Modificar el diseño del problema 31 para un registro de desplazamiento con entrada y salida serie de 16 bits.
33. Implementar un contador de 4 bits que, secuencialmente, genere el código de exceso 3 utilizando una GAL22V10 como dispositivo objeto.
34. Modificar el diseño del problema 33 para que el contador se pueda precargar en paralelo con cualquier número especificado de 4 bits en código de exceso 3.
35. Desarrollar un diagrama de estados para un sistema de control de ascensor para tres pisos.
36. Utilizando el diagrama de estados del problema 35, escribir un archivo de entrada en ABEL para implementar la lógica de control de estados del sistema del control de ascensor para tres pisos, utilizando una GAL22V10.

Localización de fallos con el EWB

37. Abrir el archivo PR11_37.EWB contenido en el CD-ROM EWB. Determinar si el registro de desplazamiento está funcionando correctamente. Si existe un fallo, intente aislarlo.
38. Abrir el archivo PR11_38.EWB contenido en el CD-ROM EWB y probar si el contador está funcionando correctamente. Si existe un fallo, intente aislarlo.
39. Abrir el archivo PR11_39.EWB contenido en el CD-ROM EWB y probar si el contador está funcionando correctamente. Si existe un fallo, intente aislarlo.
40. Abrir el archivo PR11_40.EWB contenido en el CD-ROM EWB y probar si el contador está funcionando correctamente. Si existe un fallo, intente aislarlo.

RESPUESTAS A LAS REVISIONES DE CADA SECCIÓN

SECCIÓN 11.1

1. Los modos de la GAL22V10 son combinacional y secuencial.
2. El modo se establece mediante los bits de selección S_0 y S_1.
3. La GAL16V8 dispone de dos modos combinacionales (simple y complejo) y un modo secuencial.

SECCIÓN 11.2

1. En el modo secuencial, la salida procede de un flip-flop. En el modo combinacional la salida no procede de un flip-flop.
2. Una instrucción ISTYPE se usa para declarar una salida como combinacional o secuencial.
3. El operador := indica una función secuencial, mientras que el operador = indica una operación combinacional.
4. La extensión .CLK indica que el dispositivo de registro es un flip-flop sincronizado con la señal de reloj.

SECCIÓN 11.3

1. Q0 : = A & !B # !A & B; significa que la salida Q_0 es secuencial e igual a la operación OR-exclusiva entre A y B.
2. La extensión .AR indica una puesta a cero asíncrona de un flip-flop.
3. La constante especial .c. indica un impulso de reloj.

SECCIÓN 11.4

1. Las categorías de máquinas de estado son Moore y Mealy
2. Un contador es una máquina de estados Moore.
3. Los métodos de entrada son: ecuaciones, tabla de verdad y diagramas de estados.
4. La constante especial .x. representa una condición indiferente.

SECCIÓN 11.5

1. Los latches se requieren para almacenar un cierre temporal de un interruptor.
2. Se requiere un máximo de tres impulsos de reloj para poner el sistema en el estado ESPERA1 independientemente de su estado inicial. Puesto que se usan tres impulsos de reloj, no importan las salidas para los dos primeros impulsos.
3. No se puede utilizar el método de entrada del diagrama de estados para la lógica del display porque toda la lógica es combinacional.
4. Se puede utilizar el método de entrada de la tabla de verdad para toda la lógica.

SECCIÓN 11.6

1. No hay ninguna salida no utilizada y disponible.
2. Hay seis entradas no utilizadas y disponibles.
3. Una GAL22V10 reemplaza a cinco circuitos integrados del diseño original.

RESPUESTAS A LOS PROBLEMAS RELACIONADOS DE LOS EJEMPLOS

11.1
```
Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7     PIN 22,21,20,19,18,17,16,15 ISTYPE 'reg,invert';
```

11.2
```
'Declaración de pines
Clock, Clear                pin 1,2;
SHLD                        pin 3;
D0, D1, D2, D3, D4          pin 4,5,6,7,8 ISTYPE 'reg,buffer';
Q0, Q1, Q2, Q3, Q4          pin 14,15,16,17,18 ISTYPE 'reg,buffer';

Equations
Q0 := D0;
Q1 := Q0 & SHLD # D1 & !SHLD;
Q2 := Q1 & SHLD # D2 & !SHLD;
Q3 := Q2 & SHLD # D3 & !SHLD;
Q4 := Q3 & SHLD # D4 & !SHLD;

[Q0, Q1, Q2, Q3, Q4].CLK = Clock;
[Q0, Q1, Q2, Q3, Q4].AR = !Clear;
```

11.3 La línea 1 borra el contador a 000.
La línea 2 decrementa el contador a 100.
La línea 3 decrementa el contador a 101.
La línea 4 decrementa el contador a 111.
La línea 5 decrementa el contador a 110.

La línea 6 decrementa el contador a 010.
La línea 7 decrementa el contador a 011.
La línea 8 decrementa el contador a 001.
La línea 9 decrementa el contador a 000.
La línea 10 decrementa el contador a 001.
La línea 11 decrementa el contador a 011.
La línea 12 decrementa el contador a 010.
La línea 13 decrementa el contador a 110.
La línea 14 decrementa el contador a 111.
La línea 15 decrementa el contador a 101.
La línea 16 decrementa el contador a 100.
La línea 17 decrementa el contador a 000.

11.4 La constante indiferente .x. de la primera línea permite a la tabla de verdad mostrar que las salidas se ponen a cero cuando la línea de borrado está a nivel BAJO, independientemente de los estados de Q_0, Q_1 y Q_2.

11.5 Estado A: if !Y then H else B

11.6 Las líneas de los vectores de prueba trazan los caminos, mostrados en la Figura 11.26.

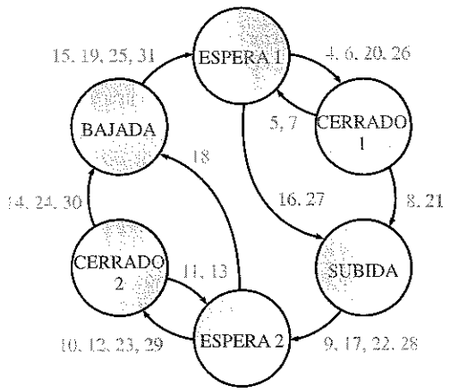


FIGURA 11.26

11.7.
```
Module Lógica_display_ascensor
Title  'Lógica del display para el sistema de ascensor para dos pisos'
"Declaración de dispositivos
   Display_ascn  DEVICE              'P16V8'
"Declaración de pines
   MOVIM, DIR              pin 2,3
   SEGADEG, SEGB, SEGC     pin 12, 13, 14    ISTYPE 'com,invert';
   FLSUB,FLBAJ             pin 18, 19        ISTYPE 'com,invert';
Equations
  FLSUB      = MOVIM & !DIR;
  FLBAJ      = MOVIM & DIR;
  SEGADEG    = !MOVIM & DIR;
  SEGB       = !MOVIM;
  SEGC       = !MOVIM & !DIR;
```



```
Test_vectors
  ([ MOVIM, DIR] -> [SEGADEG, SEGB, SEGC, FLSUB, FLBAJ])
    [   0   ,  0 ] -> [   1   ,   0 ,   0 ,   1  ,   1 ];
    [   0   ,  1 ] -> [   0   ,   1 ,   1 ,   1  ,   1 ];
    [   1   ,  0 ] -> [   1   ,   1 ,   1 ,   0  ,   1 ];
    [   1   ,  1 ] -> [   1   ,   0 ,   0 ,   1  ,   0 ];
END
```

■ RESPUESTAS AL AUTOTEST

1. (b) **2.** (b) **3.** (d) **4.** (a) **5.** (c) **6.** (d) **7.** (e) **8.** (b) **9.** (d) **10.** (c)

12

MEMORIAS

12.1 Principios de las memorias semiconductoras
12.2 Memorias de acceso aleatorio (RAM)
12.3 Memorias de sólo lectura (ROM)
12.4 Memorias ROM programables (PROM y EPROM)
12.5 Memorias flash
12.6 Expansión de memorias
12.7 Tipos especiales de memorias
12.8 Memorias ópticas y magnéticas
12.9 Pruebas y localización de averías
12.10 Aplicación a los sistemas digitales

Simulación por computadora de circuitos digitales:
El tutorial de Electronics Workbench (EWB) se encuentra en la dirección http://www.prenhall.com/floyd.

■ OBJETIVOS DEL CAPÍTULO

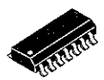
- ❑ Definir las características básicas de las memorias.
- ❑ Explicar qué es una memoria RAM y cómo funciona.
- ❑ Explicar la diferencia entre RAM estática (SRAM, Static RAM) y RAM dinámica (DRAM, Dynamic RAM).
- ❑ Explicar qué es una memoria ROM y cómo funciona.
- ❑ Explicar cómo se programa una ROM.
- ❑ Describir los distintos tipos de PROM.
- ❑ Describir los distintos tipos de memorias flash.
- ❑ Describir la expansión de las memorias ROM y RAM para aumentar la longitud y capacidad de palabra.
- ❑ Describir la organización básica de las cintas y discos magnéticos.
- ❑ Describir el funcionamiento básico de los discos magneto-ópticos y de los discos ópticos.
- ❑ Describir los métodos básicos para probar las memorias.
- ❑ Desarrollar diagramas de flujo para probar las memorias.
- ❑ Aplicar las memorias en un sistema digital.

■ INTRODUCCIÓN

En el Capítulo 10 se han tratado los registros de desplazamiento, que son un tipo de dispositivos de almacenamiento; de hecho, un registro de desplazamiento es, esencialmente, una memoria a pequeña escala. Los dispositivos de memoria que se cubren en este capítulo se utilizan generalmente para almacenamiento a más largo plazo y de cantidades más grandes de datos de lo que los registros son capaces de permitir.

Las computadoras y otros tipos de sistemas requieren el almacenamiento permanente o semipermanente de un gran número de datos binarios. Los sistemas que están basados en microprocesadores necesitan de los dispositivos de almacenamiento y de las memorias para su funcionamiento, debido a la necesidad de almacenar los programas y mantener los datos generados durante el procesamiento.

En la terminología informática, normalmente el término *memoria* hace referencia a las memorias RAM y ROM y el término *almacenamiento* hace referencia al disco duro, a los discos flexibles y al CD-ROM. En este capítulo se estudian las memorias semiconductoras, magnéticas y ópticas.

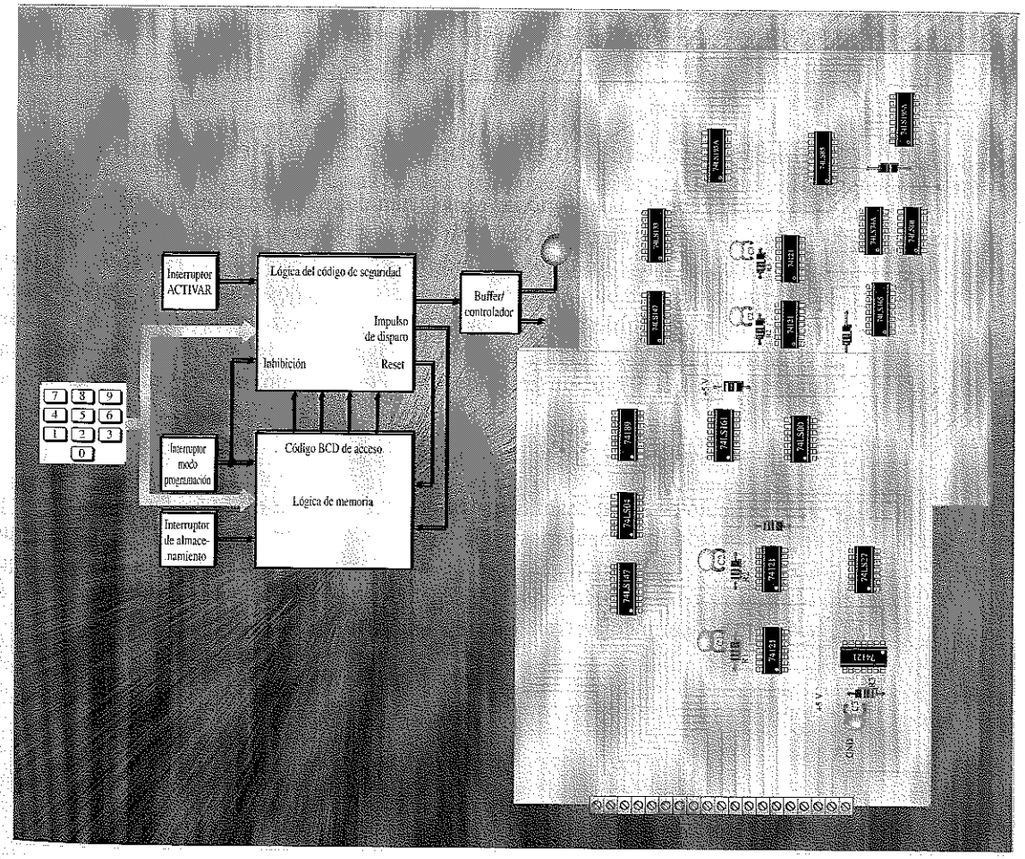


DISPOSITIVOS ESPECÍFICOS

27C16B 74F189 µPD43256B

■ APLICACIÓN A UN SISTEMA DIGITAL

Esta aplicación a un sistema digital ilustra los conceptos que se estudian en este capítulo. La aplicación de la Sección 12.10 completa el sistema de seguridad de acceso del Capítulo 10. El punto principal de este capítulo es la parte de la lógica de memoria del sistema, que almacena el código de entrada. Una vez que la lógica de memoria se haya desarrollado, se conectará con la lógica de introducción del código del Capítulo 10, para completar el sistema.



12.1 ■ PRINCIPIOS DE LAS MEMORIAS SEMICONDUCTORAS

La memoria es la parte de un sistema que almacena datos binarios en grandes cantidades. Las memorias semiconductoras están formadas por matrices de elementos de almacenamiento que pueden ser latches o condensadores.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar cómo almacena una memoria los datos binarios. ■ Exponer la organización básica de una memoria. ■ Describir la operación de escritura. ■ Describir la operación de lectura. ■ Describir la operación de direccionamiento. ■ Explicar qué son las memorias RAM y ROM.

NOTAS INFORMÁTICAS
La definición general de *palabra (word)* dice que una palabra es una unidad completa de información, consistente en una unidad de datos binarios. Cuando se aplica a las instrucciones de una computadora, una palabra se define de forma más específica como dos bytes (16 bits). Como parte muy importante del lenguaje ensamblador utilizado por las computadoras, la directiva DW (Define Word) significa que se definen datos en unidades de 16 bits. Esta definición es independiente del microprocesador o del tamaño de su bus de datos. El lenguaje ensamblador también permite definiciones de bytes (8 bits) con la directiva DB, de palabras dobles (32 bits) con la instrucción DD y de palabras cuádruples (64 bits) con la instrucción QD.

Unidades de datos binarios: bits, bytes, nibbles y palabras

Como regla general, las memorias almacenan datos en unidades que tienen de uno a ocho bits. La unidad menor de datos binarios es, como ya sabemos, el **bit**. En muchas aplicaciones, se tratan los datos en unidades de 8 bits, denominadas **bytes** o en múltiplos de unidades de 8 bits. El byte se puede dividir en dos unidades de 4 bits, que reciben el nombre de **nibbles**. Una unidad completa de información se denomina **palabra** y está formada, generalmente, por uno o más bytes. Algunas memorias almacenan datos en grupos de 9 bits; un grupo de 9 bits consta de un byte más un bit de paridad.

Matriz de memoria semiconductora básica

Cada elemento de almacenamiento en una memoria puede almacenar un 1 o un 0 y se denomina **celda**. Las memorias están formadas por matrices de celdas, como se ilustra en la Figura 12.1, en la que se utilizan 64 celdas a modo de ejemplo. Cada bloque de la **matriz de memoria** representa una celda de almacenamiento y su situación se puede especificar mediante una fila y una columna.

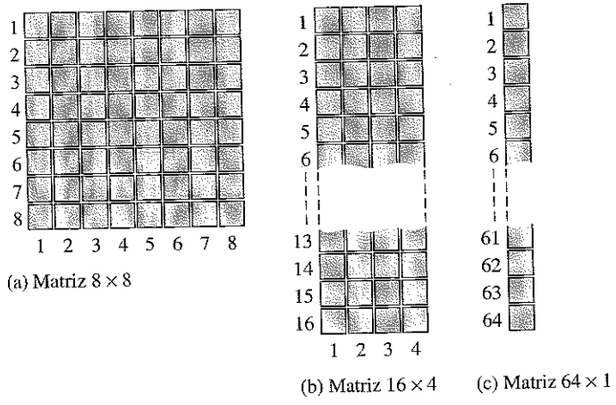
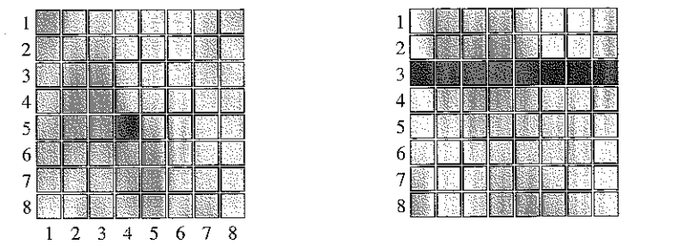


FIGURA 12.1
Matriz de almacenamiento de 64 celdas, organizada de tres formas diferentes.

La matriz de 64 celdas se puede organizar de muchas maneras en función de las unidades de datos. La Figura 12.1(a) muestra una matriz de 8 x 8, que se puede entender como una memoria de 64 bits o como una memoria de 8 bytes. La parte (b) nos muestra una matriz de 16 x 4, que es una memoria de 16 nibbles y la parte (c) presenta una matriz de 64 x 1 que es una memoria de 64 bits. Una memoria se identifica mediante el número de palabras que puede almacenar, multiplicado por el tamaño de la palabra. Por ejemplo, una memoria de 16k x 8 puede almacenar 16.384 palabras de ocho bits. La incoherencia en la expresión anterior es común en la terminología de las memorias. En realidad, el número de palabras es siempre una potencia de 2 que, en este caso, es 2^{14} = 16.384. Sin embargo, es una práctica común expresar ese número en Kbits (Kilo-bits), donde 1 Kbit = 10,24 = 2^{10} bits.

Dirección y capacidad de las memorias

La *posición* de una unidad de datos en una matriz de memoria se denomina **dirección**. Por ejemplo, en la Figura 12.2(a), la dirección de un bit en la matriz se especifica mediante la fila y columna en que está, tal como se muestra. En la Figura 12.2(b), la dirección de un byte se especifica únicamente mediante la fila. Como puede verse, la dirección depende de cómo se organice la memoria en unidades de datos. Las computadoras personales disponen de memorias organizadas en bytes. Esto significa que el grupo más pequeño de bits que se puede direccionar es ocho.



(a) La dirección del bit resaltada en gris oscuro corresponde a la fila 5, columna 4.

(b) La dirección del byte resaltada en gris oscuro corresponde a la fila 3.

FIGURA 12.2
Ejemplos de direcciones de memoria.

La **capacidad** de una memoria es el número total de unidades de datos que puede almacenar. Por ejemplo, en la matriz de memoria organizada en bits de la Figura 12.2(a), la capacidad total es de 64 bits. En la matriz de memoria organizada en bytes de la Figura 12.2(b), la capacidad es de 8 bytes, que es lo mismo que 64 bits. Típicamente, las memorias de computadora disponen de 32 Mbytes (Mbyte es megabyte), o más, de memoria interna.

Operaciones básicas de las memorias

Puesto que una memoria almacena datos binarios, los datos deben introducirse en la memoria y deben poder recuperarse cuando se necesiten. La operación de **escritura**

coloca los datos en una posición específica de la memoria y la operación de **lectura** extrae los datos de una dirección específica de memoria. La operación de direccionamiento, que forma parte tanto de la operación de lectura como de la de escritura, selecciona la dirección de memoria específica.

Las unidades de datos se introducen en la memoria durante la operación de escritura y se extraen de la memoria durante la operación de lectura a través de un conjunto de líneas que se denominan *bus de datos*. Como se indica en la Figura 12.3, el bus de datos es bidireccional, lo que significa que los datos pueden ir en cualquiera de las dos direcciones (hacia la memoria o desde la memoria). En el caso de una memoria organizada en bytes, el bus de datos tiene al menos ocho líneas, de manera que los ocho bits de una dirección seleccionada se transmiten en paralelo. En una operación de escritura o de lectura, se selecciona una dirección introduciendo un código binario, que representa la dirección deseada, en un conjunto de líneas denominado *bus de direcciones*. El código de dirección se decodifica internamente y de esa forma se selecciona la dirección adecuada. El número de líneas del bus de direcciones depende de la capacidad de la memoria. Por ejemplo, un código de dirección de 15 bits puede seleccionar 32.768 posiciones (2^{15}) en la memoria; un código de dirección de 16 bits puede seleccionar 65.536 (2^{16}) posiciones de memoria, etc. En las computadoras personales, un bus de direcciones de 32 bits puede seleccionar 4.294.967.296 (2^{32}) posiciones, lo que se expresa como 4 G.

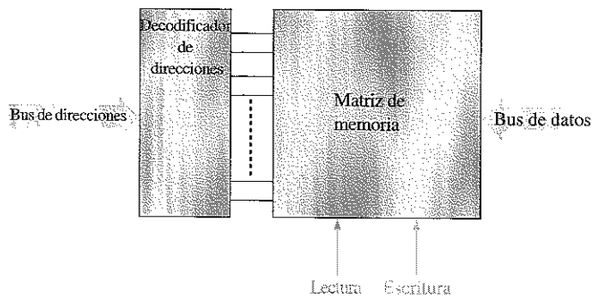
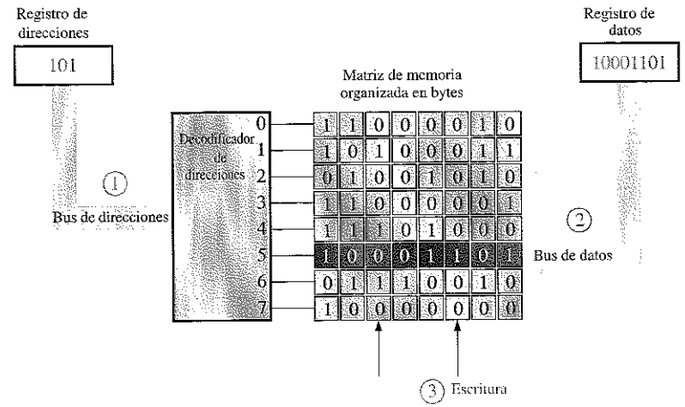


FIGURA 12.3
Diagrama de bloques de una memoria, mostrando el bus de direcciones, el decodificador de direcciones, el bus de datos y entradas de lectura/escritura.

La operación de escritura. En la Figura 12.4 se muestra la operación de escritura simplificada. Para almacenar un byte de datos en memoria, se introduce en el bus de direcciones un código que se encuentra almacenado en el registro de direcciones. Una vez que el código de dirección está ya en el bus, el decodificador de direcciones decodifica la dirección y selecciona la posición de memoria especificada. La memoria recibe entonces una orden de escritura y los datos almacenados en el registro de datos se introducen en el bus de datos, y se almacenan en la dirección de memoria especificada, completándose así la operación de escritura. Cuando se escribe un nuevo byte de datos en una dirección de memoria, se sobrescribe y destruye el byte de datos actualmente almacenado en esa dirección.



1. El código de dirección 101 se coloca en el bus de direcciones y se selecciona la dirección 5.
2. El byte de datos se coloca en el bus de datos.
3. El comando de escritura hace que el byte de datos se almacene en la dirección 5, reemplazando a los datos anteriores.

FIGURA 12.4
Ilustración de la operación de escritura.

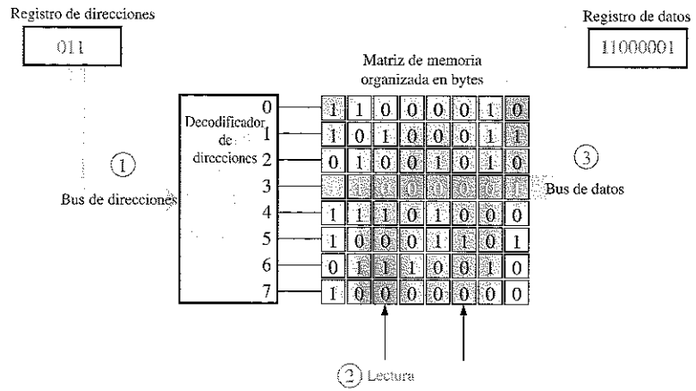
La operación de lectura. En la Figura 12.5 se muestra la operación de lectura simplificada. De nuevo, se introduce en el bus de direcciones un código almacenado en el registro de direcciones. Una vez que el código de dirección se encuentra en el bus, el decodificador de direcciones decodifica la dirección y selecciona la posición especificada de la memoria. La memoria recibe entonces una orden de lectura, y una "copia" del byte de datos almacenado en la dirección de memoria seleccionada se introduce en el bus de datos y se carga en el registro de datos, finalizando así la operación de lectura. Cuando se lee un byte de datos de una dirección de memoria, éste sigue almacenado en dicha dirección y no se destruye. Esto se denomina *lectura no destructiva*.

Las memorias RAM y ROM

Las dos principales categorías de memorias semiconductoras son las memorias RAM y ROM. La memoria **RAM** (random-access memory, memoria de acceso aleatorio) es un tipo de memoria en la que se tarda lo mismo en acceder a cualquier dirección de memoria y éstas se pueden seleccionar en cualquier orden, tanto en una operación de lectura como de escritura. Todas las RAM poseen la capacidad de *lectura* y *escritura*. Debido a que las memorias RAM pierden los datos almacenados cuando se desconecta la alimentación, reciben el nombre de memorias **volátiles**.

La memoria **ROM** (read-only memory, memoria de sólo lectura) es un tipo de memoria en la que los datos se almacenan de forma permanente o semipermanente. Los datos se pueden leer de una ROM, pero no existe la operación de escritura como en las RAM. La ROM, al igual que la RAM, es una memoria de acceso aleatorio pero, tradicionalmente, el término RAM se reserva para las memorias de acceso aleatorio de *lectura/escritura*. En este capítulo veremos varios tipos de memorias RAM y

ROM. Debido a que las ROM mantienen los datos almacenados incluso si se desconecta la alimentación, reciben el nombre de memorias **no volátiles**.



(1) El código de dirección 011 se coloca en el bus de direcciones y se selecciona la dirección 3.
(2) Se aplica el comando de lectura.
(3) El contenido de la dirección 3 se coloca en el bus de datos y se desplaza al registro. El contenido de la dirección 3 no se destruye a consecuencia de la operación de lectura.

FIGURA12.5
Ilustración de la operación de lectura.

REVISIÓN DE LA SECCIÓN 12.1

1. ¿Cuál es la unidad más pequeña de datos que se puede almacenar en una memoria?
2. ¿Cuál es la capacidad en bits de una memoria que puede almacenar 256 bytes de datos?
3. ¿En qué consiste la operación de escritura?
4. ¿En qué consiste la operación de lectura?
5. ¿Cómo se localiza una determinada unidad de datos en una memoria?
6. Describir las diferencias entre una RAM y una ROM.

12.2 ■ MEMORIAS DE ACCESO ALEATORIO (RAM)

Las RAM son memorias de lectura-escritura en las que los datos se pueden escribir o leer en cualquier dirección seleccionada en cualquier orden. Cuando se escriben los datos en una determinada dirección de la RAM, los datos almacenados previamente son reemplazados por la nueva unidad de datos. Cuando una unidad de datos se lee de una determinada dirección de la RAM, los datos de esa dirección permanecen almacenados y no son destruidos por la operación de lectura. Esta operación no destructiva de lectura se puede entender como una copia del contenido de una dirección, dejando dicho contenido intacto. La RAM se utiliza habitualmente para almacenamiento de datos a corto plazo, ya que no puede conservar los datos almacenados cuando se desconecta la alimentación.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Nombrar las dos categorías de RAM. ■ Explicar qué es una SRAM. ■ Describir la celda de almacenamiento de una SRAM. ■ Explicar la diferencia entre una SRAM asíncrona y una SRAM de ráfaga síncrona. ■ Explicar qué es una DRAM. ■ Describir la celda de almacenamiento de una DRAM. ■ Explicar los tipos de DRAM. ■ Comparar la SRAM con la DRAM.

La familia de memorias RAM

Las dos categorías de memorias RAM son la *RAM estática* (SRAM) y la *RAM dinámica* (DRAM). Las RAM estáticas utilizan flip-flops como elementos de almacenamiento y, por tanto, pueden almacenar datos de forma indefinida *siempre que se aplique una alimentación continua.* Las RAM dinámicas utilizan condensadores como elemento de almacenamiento y no pueden mantener los datos mucho tiempo sin recargar los condensadores mediante el proceso de **refresco**. Tanto las SRAM como las DRAM perderán los datos cuando se elimine la alimentación continua, por lo que se clasifican como memorias volátiles.

Los datos pueden leerse mucho más rápidamente en una SRAM que en una DRAM. Sin embargo, las DRAM pueden almacenar muchos más datos que las SRAM para un tamaño físico y coste dados, ya que la celda de las DRAM es mucho más sencilla y se pueden incluir muchas más celdas en un área determinada que en una memoria SRAM. Los tipos básicos de memorias SRAM son las memorias *SRAM asíncronas* y las *SRAM síncronas de ráfaga*. Los tipos básicos de DRAM son la *DRAM con modo página rápido (Fast Page Mode, FPM DRAM)*, la *DRAM con salida de datos extendida (Extended Data Output, EDO DRAM)*, la *DRAM con salida de datos extendida en ráfaga (Burst Extended Data Output, BEDO DRAM)* y la *DRAM síncrona (Synchronous, SDRAM)*. Todas ellas se muestran en la Figura 12.6.

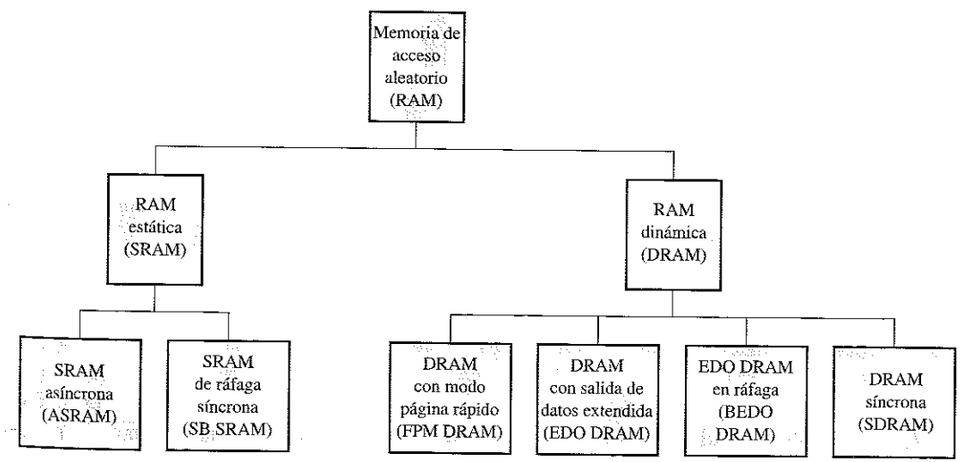


FIGURA 12.6
La familia de memorias RAM.

RAM estática (SRAM)

Celda de almacenamiento. Todas las RAM estáticas se caracterizan por las celdas de almacenamiento flip-flop que, típicamente, se implementan en circuitos integrados con varios transistores MOS (**MOSFET**). Existen también algunas SRAM más pequeñas implementadas con transistores bipolares. Cuando se aplica alimentación continua a una celda de **memoria estática** se puede mantener un estado 1 o 0 indefinidamente. Si se retira la alimentación, el bit de datos almacenado se perderá.

La Figura 12.7 muestra una celda básica de almacenamiento de tipo flip-flop para la **SRAM**. La celda se selecciona mediante un nivel activo en la línea Selección de bit y un bit de datos (1 o 0) se escribe en la celda colocándolo en las líneas Datos y $\overline{\text{Datos}}$. Un bit de datos se puede leer extrayéndolo de las líneas Datos y $\overline{\text{Datos}}$. Los datos de entrada y los datos de salida pueden compartir las mismas líneas porque las operaciones de escritura y lectura se producen en diferentes instantes.

Matriz básica de celdas de memoria estáticas. Las celdas de almacenamiento en una SRAM se organizan en filas y columnas, tal y como se ilustra en la Figura 12.8 para el caso de una matriz *n x 4*. Todas las celdas de una misma fila comparten la misma línea Selección Fila. Cada conjunto de líneas Datos y $\overline{\text{Datos}}$ van a cada celda situada en una determinada columna y se conectan a una única línea de datos, que sirve como entrada y salida (E/S datos), a través de los buffers de entrada y salida de datos.

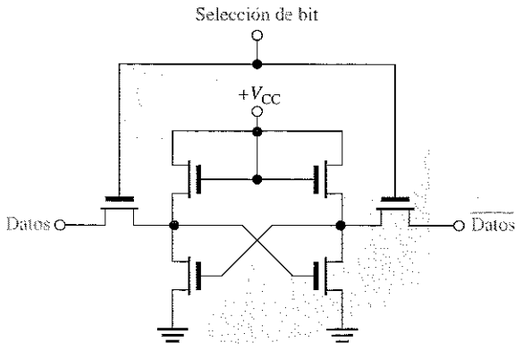


FIGURA 12.7
Célula típica de almacenamiento de una RAM estática, que muestra símbolos simplificados de transistor.

Para escribir una unidad de datos, en este caso un grupo de 4 bits, en una fila de celdas determinada de la matriz de memoria, la línea Selección Fila se pone en su estado activo y los 4 bits de datos se colocan en las líneas de entrada de datos. La línea de escritura (write) se pone entonces en estado activo, lo que da lugar a que cada bit de datos se almacene en una celda seleccionada en la columna asociada. Para leer una unidad de datos, se pone en estado activo la línea de lectura (read), lo que hace que los 4 bits de datos almacenados en la fila seleccionada aparezcan en las líneas de salida de datos.

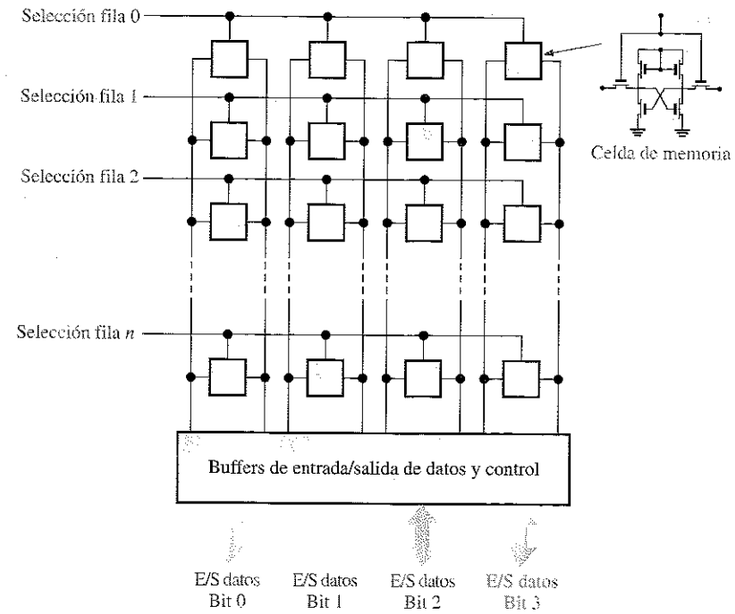


FIGURA 12.8
Matriz básica de la memoria SRAM.

Organización de la SRAM asíncrona básica

Una SRAM asíncrona es aquella en la que su funcionamiento no está sincronizado con un reloj de sistema. Para ilustrar la organización general de una SRAM, vamos a utilizar una memoria de 32k x 8 bits, tal como el circuito integrado µPD43256B. En la Figura 12.9 se muestra el símbolo lógico de esta memoria.

En el modo de lectura (READ), los 8 bits de datos que se almacenan en una dirección determinada aparecen en las líneas de salida de datos. En el modo de escritura (WRITE), los 8 bits de datos que se aplican a las líneas de entrada de datos se almacenan en la dirección seleccionada. Las líneas de entrada y salida de datos (E/S_1 a E/S_8) son las mismas líneas. Durante la operación de lectura, éstas actúan como líneas de salida (S_1 a S_8) y durante la operación de escritura actúan como líneas de entrada (E_1 a E_8).

Salidas triestado y buses. Los buffers triestado en una memoria permiten que las líneas de datos actúen como líneas de entrada o salida y conectan la memoria con el bus de datos en una computadora. Estos buffers tienen tres posibles estados de salida: ALTO (1), BAJO (0) y ALTA-Z (alta impedancia, abierto). Las salidas triestado se indican en los símbolos lógicos mediante un pequeño triángulo invertido (▽), como se muestra en la Figura 12.9, y se utilizan por compatibilidad con las estructuras de bus, como las que se encuentran en los sistemas basados en microprocesador.

Físicamente, un **bus** es un conjunto de caminos conductores que sirven para interconectar dos o más componentes funcionales de un sistema o de varios sistemas diferentes. Eléctricamente, un bus es una colección de señales y de niveles de tensión

y/o corriente específicos que permiten a los distintos dispositivos conectados al bus comunicarse y funcionar correctamente.

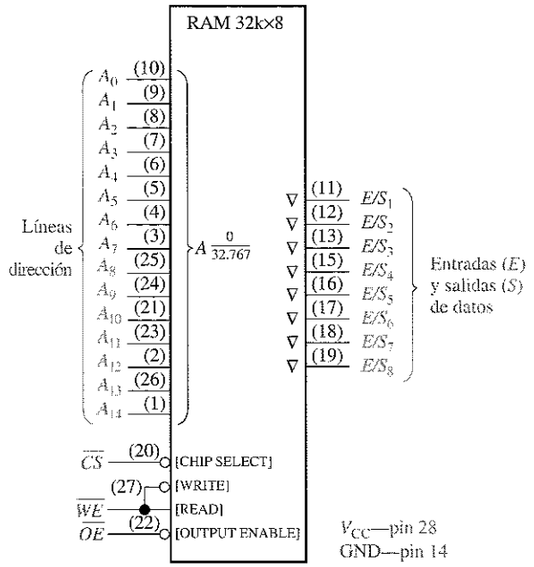


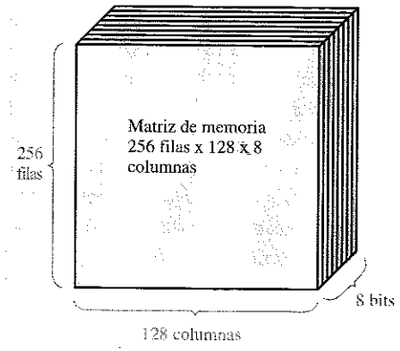
FIGURA 12.9
Diagrama lógico de una SRAM de 32k × 8.

Por ejemplo, un microprocesador se conecta a las memorias y dispositivos de entrada/salida mediante determinadas estructuras de bus. Un bus de direcciones permite al microprocesador direccionar las memorias, y el bus de datos sirve para la transferencia de datos entre el microprocesador, las memorias y los dispositivos de entrada/salida, tales como monitores, impresoras, teclados y modems. El bus de control permite al microprocesador controlar la temporización y la transferencia de datos entre los distintos componentes.

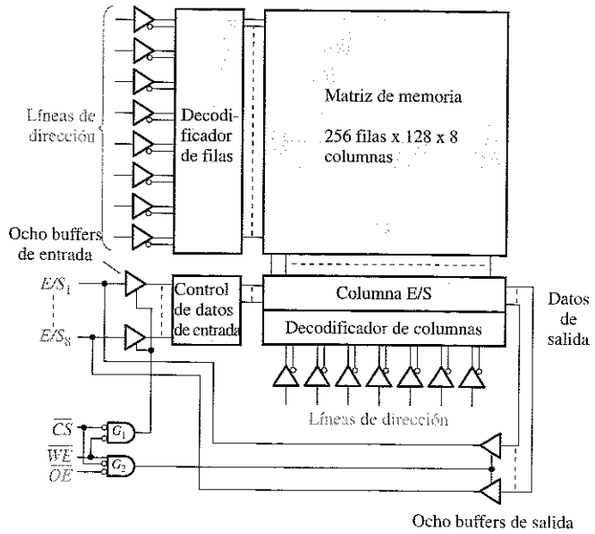
Matriz de memoria. Los chips SRAM se pueden organizar en bits, nibbles (4 bits), bytes (8 bits) o múltiplos de bytes (16, 24 o 32 bits).

La Figura 12.10 muestra la organización de una SRAM típica de 32k × 8. La matriz de celdas de memoria está organizada en 256 filas y 128 x 8 columnas (128 columnas, cada una de 8 bits), como se muestra en la parte (a) de la figura. En la práctica, dispone de 2^{15} = 32.768 direcciones, y cada dirección contiene 8 bits. La capacidad de esta memoria ejemplo es entonces de 32.768 bytes (lo que normalmente se expresa como 32 kbytes).

La SRAM de la Figura 12.10(b) trabaja del siguiente modo. En primer lugar, la entrada de habilitación del chip, $\overline{CS}$, debe estar a nivel BAJO para que la memoria funcione. Ocho de las quince líneas de dirección se decodifican en el decodificador de filas, de modo que se selecciona una de las 256 filas. Las restantes siete líneas de dirección las decodifica el decodificador de columnas, de modo que se selecciona una de las 128 columnas de 8 bits.



(a) Configuración de la matriz de memoria



(b) Diagramas de bloques de la memoria

FIGURA 12.10
Organización básica de una SRAM de 32k × 8.

LECTURA. En el modo lectura (READ), la entrada de habilitación de escritura $\overline{WE}$ está a nivel ALTO y la salida de habilitación $\overline{OE}$ está a nivel BAJO. La puerta G_1 desactiva los buffers de entrada, y la puerta G_2 activa los buffers de tres estados de salida de las columnas. Por tanto, los ocho bits de datos almacenados en la dirección seleccionada se llevan a través de las E/S de las columnas hasta las líneas de datos (E/S_1 a E/S_8), que actúan como líneas de salida de datos.

ESCRITURA. En el modo escritura (WRITE), $\overline{WE}$ está a nivel BAJO y $\overline{OE}$ está a nivel ALTO. La puerta G_1 activa los buffers de entrada, y la puerta G_2 desactiva los buffers de salida. Por tanto, los ocho bits de datos de entrada de las líneas de datos se llevan a través del control de datos de entrada y de la E/S de columna a la dirección seleccionada, y se almacenan.

Ciclos de lectura y escritura. La Figura 12.11 muestra un diagrama de tiempos típico para un ciclo de lectura y un ciclo de escritura de una memoria. En la parte (a) se presenta el ciclo de lectura; se aplica un código de dirección válido a las líneas de dirección durante un intervalo de tiempo especificado, que se denomina *periodo del ciclo de lectura,* t_{RC}. A continuación, las entradas de habilitación del chip ($\overline{CS}$) y de habilitación de salida ($\overline{OE}$) pasan a nivel BAJO. Un intervalo de tiempo después de que la entrada $\overline{OE}$ haya pasado a nivel BAJO, un byte de datos válido procedente de la dirección seleccionada se presenta en las líneas de datos. Este intervalo de tiempo se denomina *tiempo de acceso de la habilitación de salida,* t_{GQ}.

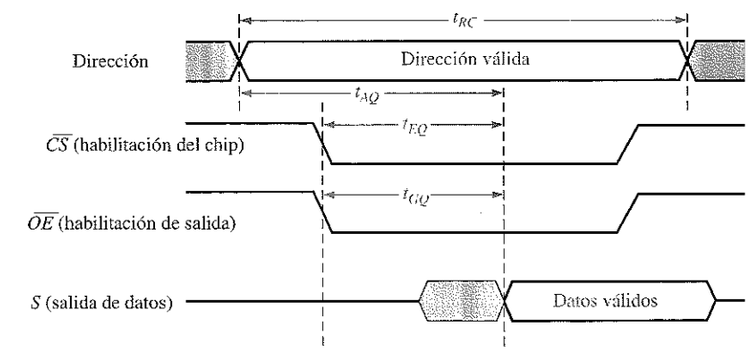
Existen otros dos tiempos de acceso en el ciclo de lectura: el *tiempo de acceso de dirección,* t_{AQ}, que se mide desde el principio de una dirección valida hasta que los datos válidos aparecen en las líneas de datos, y el *tiempo de acceso de la habilitación*

del chip, t_{EQ}, que se mide desde la transición de nivel alto a nivel bajo de $\overline{CS}$ hasta que los datos válidos aparecen en las líneas de datos.

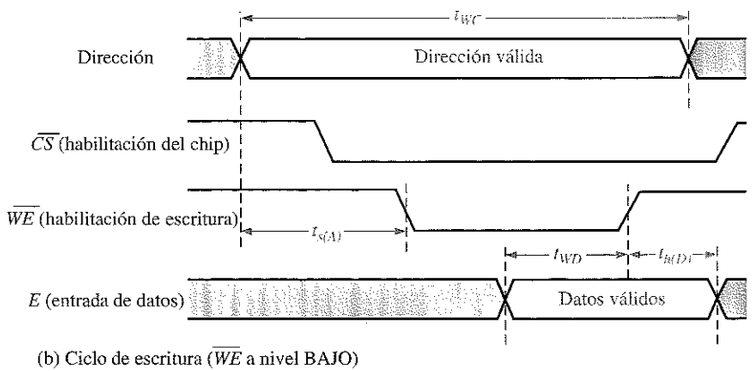
En cada ciclo de lectura, se lee de la memoria una unidad de datos, en este caso un byte.

La Figura 12.11(b) muestra el ciclo de escritura. Un código de dirección válido se aplica a las líneas de dirección durante un intervalo de tiempo especificado que se denomina *periodo del ciclo de escritura,* t_{WC}. A continuación, las entradas de habilitación del chip ($\overline{CS}$) y de habilitación de escritura ($\overline{WE}$) pasan a nivel BAJO. El intervalo de tiempo requerido desde el inicio de una dirección válida hasta que la entrada $\overline{WE}$ pasa a nivel BAJO se denomina *tiempo de establecimiento de dirección,* $t_{S(A)}$. El tiempo que la entrada $\overline{WE}$ debe estar a nivel BAJO es la anchura del impulso de escritura. El tiempo que la entrada $\overline{WE}$ debe permanecer a nivel BAJO después de haber aplicado los datos válidos en las entradas de datos se designa por t_{WD}. El tiempo que los datos de entrada válidos deben permanecer en las líneas de entrada después de que la entrada $\overline{WE}$ pase a nivel ALTO es el *tiempo de mantenimiento de datos,* $t_{h(D)}$

En cada ciclo de escritura, se escribe en la memoria una unidad de datos.



(a) Ciclo de lectura ($\overline{WE}$ a nivel ALTO)



(b) Ciclo de escritura ($\overline{WE}$ a nivel BAJO)

FIGURA 12.11
Diagramas de tiempos de los ciclos de lectura y escritura básicos para la SRAM de la Figura 12.10.

Organización básica de la SRAM síncrona de ráfaga

A diferencia de la SRAM asíncrona, una SRAM síncrona está sincronizada con el reloj del sistema. Por ejemplo, en un sistema informático, la SRAM síncrona opera con la misma señal de reloj que el microprocesador, de modo que el microprocesador y la memoria están sincronizados para conseguir una operación más rápida.

El concepto fundamental en que se basa la naturaleza síncrona de una SRAM puede ilustrarse con la Figura 12.12, que es un diagrama de bloques simplificado de una memoria de 32k × 8, con propósitos ilustrativos. La SRAM síncrona es muy similar a la SRAM asíncrona, en términos de la matriz de memoria, del decodificador de direcciones y de las entradas de lectura/escritura y activación. La diferencia fundamental es que la SRAM síncrona utiliza registros con señal de reloj para sincronizar todas las entradas con el reloj del sistema. Tanto la dirección, como la entrada de lectura/escritura, la señal de activación del chip y los datos de entrada se enclavan en sus respectivos registros con el flanco activo del pulso de reloj. Una vez enclavada esta información, la operación de la memoria estará en sincronía con el reloj.

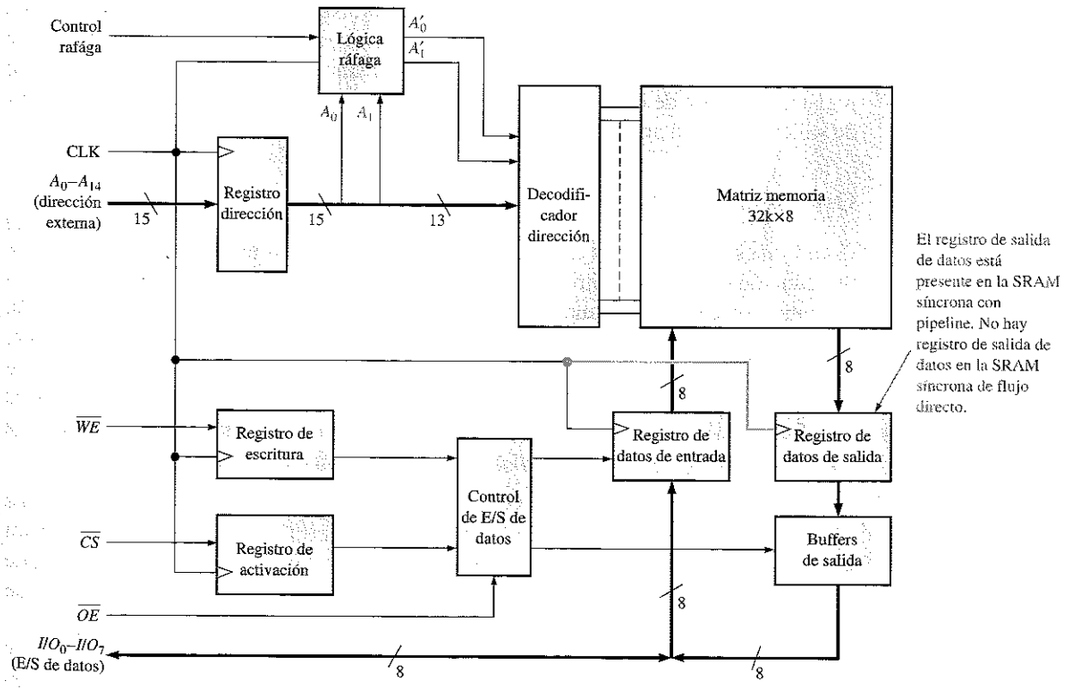
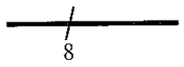


FIGURA 12.12
Diagrama de bloques básico de una SRAM de ráfaga síncrona.

Para simplificar, se presenta en la Figura 12.12 una notación para expresar múltiples líneas paralelas o líneas de bus, como alternativa a dibujar cada línea por separado. Un conjunto de líneas paralelas puede expresarse mediante una sola línea grue-

sa atravesada por una barra y con una indicación del número de líneas distintas que forman el conjunto. Por ejemplo, la siguiente notación representa un conjunto de ocho líneas paralelas:



Los bits de dirección A_0 a A_{14} se enclavan en el registro de dirección con el flanco positivo de un pulso de reloj. En el mismo pulso de reloj, el estado de las líneas de activación de escritura ($\overline{WE}$) y de selección de chip ($\overline{CS}$) se enclava en el registro de escritura y el registro de activación, respectivamente. Éstos son registros de un único bit o simplemente flip-flops. Asimismo, los datos de entrada son enclavados con el mismo pulso de reloj en el registro de datos de entrada para las operaciones de escritura, y los datos existentes en una dirección de memoria seleccionada se enclavan en el registro de salida de datos para las operaciones de lectura, según determine el control de E/S de datos, basándose en las entradas procedentes del registro de escritura, del registro de activación y de la línea de activación de salida ($\overline{OE}$).

Existen dos tipos básicos de memoria SRAM síncronas: de *flujo directo* y con *pipeline*. La SRAM síncrona de flujo directo no dispone de un registro de salida de datos, por lo que los datos de salida fluyen asíncronamente hacia las líneas de E/S de datos a través de los búferes de salida. La SRAM síncrona con **pipeline** dispone de un registro de salida de datos, como se muestra en la Figura 12.12, por lo que los datos de salida se presentan síncronamente en las líneas de E/S de datos.

Operación en modo ráfaga

Como muestra la Figura 12.12, las memorias SRAM síncronas tienen normalmente una función de ráfaga de direcciones, que permite a la memoria leer o escribir en hasta cuatro posiciones utilizando una única dirección. Cuando se enclava una dirección externa en el registro de direcciones, los dos bits menos significativos de la dirección, A_0 y A_1, se aplican al circuito de la lógica de ráfaga. Éste produce una secuencia de cuatro direcciones internas añadiendo 00, 01, 10 y 11 a los dos bits de dirección menos significativos en sucesivos pulsos de reloj. La secuencia comienza siempre con la dirección base, que es la dirección externa almacenada en el registro de dirección.

Lógica de ráfaga. La lógica de la ráfaga de direcciones en una SRAM síncrona típica está compuesta por un contador binario y puertas OR-exclusiva, como muestra la Figura 12.13. Para una lógica de ráfaga de 2 bits, la secuencia interna de ráfaga de direcciones se forma a partir de los bits A_2-A_{14} de la dirección base, más los dos bits de la dirección de ráfaga, A'_1 y A'_0.

Al comenzar la secuencia de ráfaga, el contador se encuentra en su estado 00 y los dos bits menos significativos de la dirección se aplican a las entradas de las puertas XOR. Suponiendo que tanto A_0 como A_1 sean 0, los dos bits menos significativos de la secuencia de direcciones interna serían 00, 01, 10 y 11.

Memoria caché

Una de las principales aplicaciones de las memorias SRAM es la implementación de memorias caché en computadoras. La **memoria caché** es una memoria de alta velocidad y relativamente pequeña que almacena los datos o instrucciones más recientemente utilizados de la memoria principal, más grande pero más lenta. La memoria principal utiliza RAM dinámica (DRAM), de la que hablaremos a continuación. Normalmente, la memoria SRAM es varias veces más rápida que la memoria DRAM. En conjunto, la memoria caché hace que el microprocesador pueda acceder a la información almacenada mucho más rápido que si sólo se empleara memoria DRAM de alta capacidad. La memoria caché es, básicamente, un método eficiente en términos de coste para mejorar el rendimiento del sistema sin tener que incurrir en el gasto de hacer que toda la memoria sea más rápida.

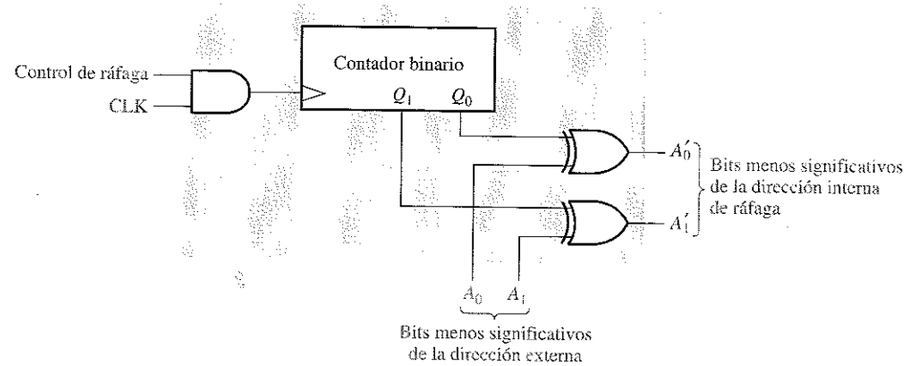


FIGURA 12.13
Lógica de la ráfaga de direcciones.

El concepto de memoria caché se basa en la idea de que los programas informáticos tienden a obtener instrucciones o datos de un área de la memoria principal antes de pasar a otra área. Básicamente, el controlador de la caché "adivina" qué área de la lenta memoria dinámica necesitará a continuación la unidad central de proceso (CPU), y mueve el contenido de dicha área a la memoria caché, para que esté listo cuando sea necesario. Si el controlador de caché ha realizado una estimación correcta, los datos están disponibles de manera inmediata para el microprocesador. Si la estimación del controlador de caché es errónea, la CPU debe acudir a la memoria principal y esperar mucho más tiempo para obtener las instrucciones o datos correctos. Afortunadamente, el controlador de caché tiene razón la mayor parte de las veces.

Visión metafórica de la caché. Hay muchas analogías que pueden usarse para describir una memoria caché, pero tal vez lo más efectivo sea compararla con una nevera doméstica. Una nevera doméstica puede considerarse como una especie de "caché" para determinados productos alimenticios, mientras que el supermercado es la memoria principal donde se almacena toda la comida. Cada vez que deseamos comer o beber algo, podemos ir primero a la nevera (caché), para ver si contiene el producto que buscamos. Si es así, nos ahorramos un montón de tiempo. Si el producto no se encuentra allí, tendremos que invertir un tiempo adicional en obtenerlo del supermercado.

Cachés L1 y L2. Las cachés de nivel 1 (caché L1) están usualmente integradas en el chip del procesador y tienen una capacidad de almacenamiento muy limitada. La caché L1 se conoce también con el nombre de caché *primaria*. Una caché de nivel 2 (caché L2) es un chip o conjunto de chips de memoria independiente, externo al procesador, y usualmente dispone de una capacidad de almacenamiento mayor que una caché L1. La caché L2 también se conoce con el nombre de caché *secundaria*. Algunos sistemas pueden tener cachés de nivel superior (L3, L4, etc.), pero L1 y L2 son los más comunes. Asimismo, algunos sistemas emplean una caché de disco, ubicada en la memoria DRAM principal y utilizada para mejorar el rendimiento del disco duro, porque la DRAM, aunque mucho más lenta que la memoria SRAM, sigue siendo mucho más rápida que la unidad de disco duro. La Figura 12.14 ilustra dos memorias caché L1 y L2 en un sistema informático.

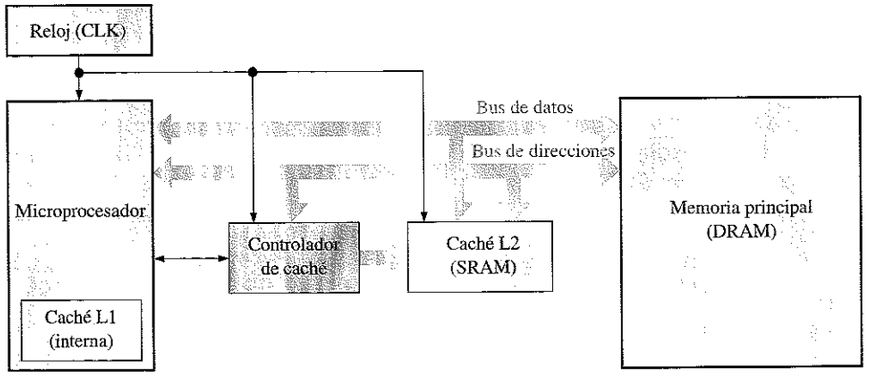


FIGURA 12.14
Diagrama de bloques mostrando memorias caché L1 y L2 en un sistema informático.

Las celdas de almacenamiento de la RAM dinámica (DRAM)

Las celdas de las **memorias dinámicas** almacenan un bit de datos en un condensador en lugar de en un latch. La ventaja de este tipo de celda es que es muy sencilla, lo que permite construir matrices de memoria muy grandes en un chip, a un coste por bit más bajo que el de las memorias estáticas. La desventaja es que el condensador de almacenamiento no puede mantenerse cargado más que un periodo de tiempo, y el dato almacenado se pierde a no ser que su carga se refresque periódicamente. La operación de refresco requiere circuitería de memoria adicional y complica el funcionamiento de la **DRAM**. La Figura 12.15 presenta una celda típica de una DRAM, formada pór un único transistor MOS (MOSFET) y un condensador.

En este tipo de celda, el transistor actúa como un interruptor. El funcionamiento básico simplificado se ilustra en la Figura 12.16, y es el siguiente. Un nivel BAJO en la línea $R/\overline{W}$ (modo escritura) activa el buffer de tres estados de entrada y desactiva el buffer de salida. Para escribir un 1 en la celda, la línea D_{IN} debe estar a nivel ALTO, y el transistor debe ser puesto en conducción poniendo un nivel ALTO en la línea de fila. El transistor opera como un interruptor cerrado, que conecta el condensador a la

línea de bit. Esta conexión permite al condensador cargarse con una tensión positiva, como muestra la Figura 12.16(a). Cuando se almacena un 0, se aplica un nivel BAJO a la línea D_{IN}. Si el condensador almacenaba un 0, permanece descargado; ahora bien, si almacenaba un 1, se descarga como se indica en la Figura 12.16(b). Cuando la línea de fila vuelve al nivel BAJO, el transistor no conduce y desconecta el condensador de la línea de bit, con lo que la carga (1 o 0) "queda atrapada" en el condensador.

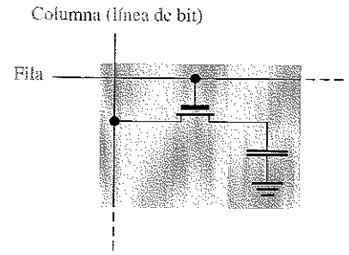
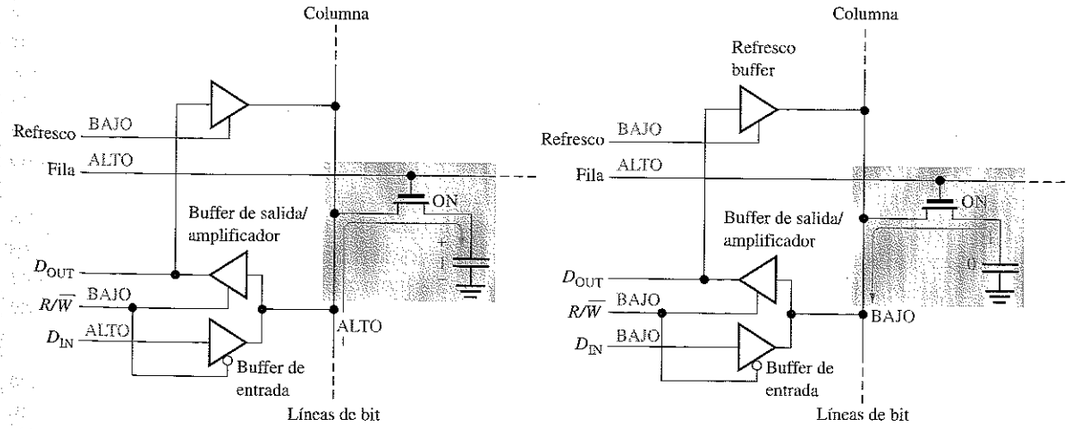


FIGURA 12.15
Celda de una RAM dinámica MOS.

Para leer una celda, la línea $R/\overline{W}$ (Read/$\overline{\text{Write}}$, lectura/$\overline{\text{escritura}}$) se pone a nivel ALTO, lo que activa el buffer de salida y desactiva el buffer de entrada. Cuando la línea de fila se pone a nivel ALTO, el transistor conduce y conecta el condensador a la línea de bit y, por tanto, al buffer de salida (amplificador). De esta manera, el bit de datos aparece en la línea de salida de datos (D_{OUT}). Este proceso se ilustra en la Figura 12.16(c).



(a) Escritura de un 1 en la celda de memoria

(b) Escritura de un 0 en la celda de memoria

FIGURA 12.16
Funcionamiento básico de una celda de una DRAM.

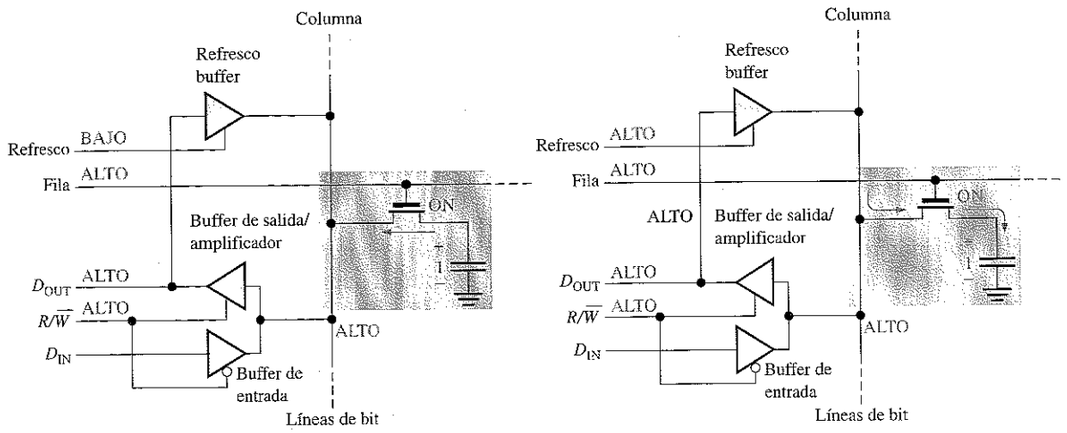


FIGURA 12.16 ***(continuación)***
Funcionamiento básico de una celda de una DRAM.

Para refrescar la celda de memoria, la línea $R/\overline{W}$, la línea de fila y la línea de refresco se ponen a nivel ALTO. El transistor conduce, conectando el condensador a la línea de bit. El buffer de salida se activa y el bit de datos almacenado se aplica a la entrada del buffer de refresco, el cual se activa mediante el nivel ALTO de la entrada de refresco. Esto da lugar a una tensión en la línea de bit que corresponde al bit almacenado, recargando el condensador, como se ilustra en la Figura 12.16(d).

Organización básica de una RAM dinámica (DRAM)

La principal aplicación de las DRAM se encuentra en la memoria principal de las computadoras. La diferencia principal entre las DRAM y SRAM es el tipo de celda de memoria. Como se ha visto, la celda de la memoria DRAM está formada por un transistor y un condensador, y es mucho más sencilla que la celda de la SRAM. Esto permite densidades mucho mayores en las DRAM, lo que da lugar a mayores capacidades de bits para una determinada área de chip, aunque el tiempo de acceso es mucho mayor.

De nuevo, dado que la carga almacenada en un condensador tiende a perderse, las celdas de una DRAM requieren una operación de refresco frecuente para conservar los bits de datos almacenados. Este requisito da lugar a una circuitería más compleja que en la SRAM. A continuación se exponen varias funciones comunes en la mayor parte de las DRAM, utilizando como ejemplo una DRAM genérica de 1 M x 1 bit.

Multiplexación de direcciones. Las DRAM utilizan una técnica denominada *multiplexación de direcciones* que sirve para reducir el número de líneas de dirección. La Figura 12.17 muestra el diagrama de bloques de una DRAM de 1.048.576 bits (1 Mbit) con una organización de 1 M x 1. Vamos a centrarnos en los bloques de color gris oscuro para ilustrar la multiplexación de direcciones. Los bloques en color gris claro representan la lógica de refresco.

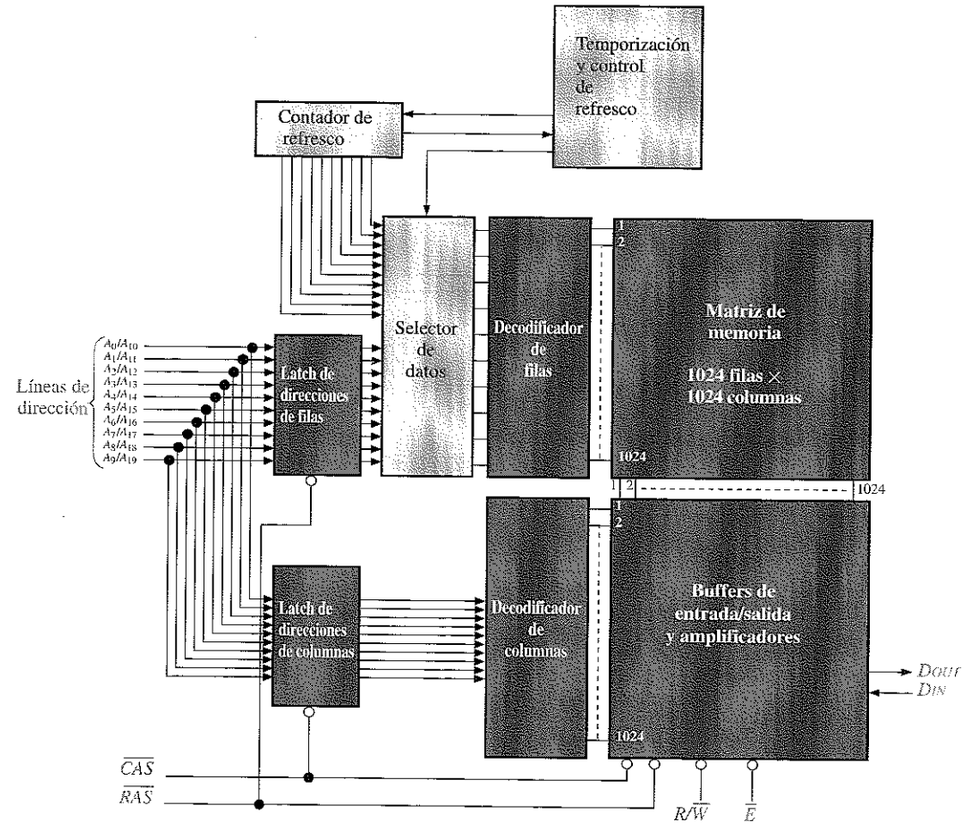


FIGURA 12.17
Diagrama de bloques simplificado de una DRAM de 1M × 1.

Las diez líneas de dirección se multiplexan en el tiempo al comienzo de un ciclo de memoria mediante la validación de dirección de fila ($\overline{RAS}$) y la validación de dirección de columna ($\overline{CAS}$), en dos campos de dirección separados. En primer lugar, la dirección de fila de 10 bits se pasa al latch de direcciones de fila. Después, la dirección de columna de 10 bits se pasa al latch de direcciones de columna. Las direcciones de fila y columna se decodifican para seleccionar una de las 1.048.576 (2^{20} = 1.048.576) direcciones de la matriz de memoria. En la Figura 12.18 se presenta el diagrama de tiempos básico para la operación de multiplexación de direcciones.

Ciclos de lectura y escritura. Al inicio de cada ciclo de memoria de lectura o escritura, $\overline{RAS}$ y $\overline{CAS}$ se activan (nivel BAJO) para multiplexar las direcciones de fila y columna hacia los latches y decodificadores. Durante el ciclo de lectura, la entrada $R/\overline{W}$ está a nivel ALTO. Durante el ciclo de escritura, la entrada $R/\overline{W}$ está a nivel BAJO. Esto se ilustra en la Figura 12.19.

Ciclos del modo página. En los ciclos de lectura y escritura normales descritos anteriormente, primero se carga la dirección de fila de una posición de memoria con-

creta mediante la entrada activa a nivel BAJO $\overline{RAS}$, y luego se carga la dirección de columna de esa posición mediante la entrada activa a nivel BAJO $\overline{CAS}$. Después se selecciona la siguiente posición mediante otra entrada $\overline{RAS}$ seguida de $\overline{CAS}$, y así sucesivamente.

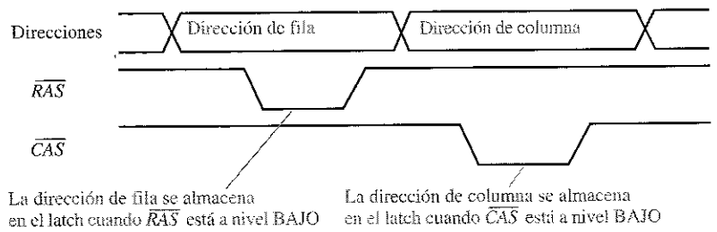


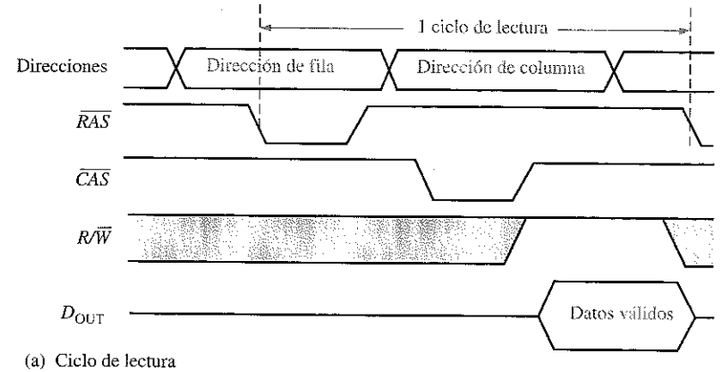
FIGURA 12.18
Diagrama de tiempos básico para la multiplexación de direcciones.

Una "página" es una sección de memoria disponible en una misma dirección de fila y que consta de todas las columnas de dicha fila. El modo página rápido permite operaciones de lectura y escritura sucesivas en cada una de las direcciones de columna de una fila seleccionada. En primer lugar, se carga una dirección de fila pasando la entrada $\overline{RAS}$ a nivel BAJO y permaneciendo en este nivel, mientras que $\overline{CAS}$ bascula entre los niveles ALTO y BAJO. Sólo se selecciona una dirección de fila, que permanece seleccionada mientras que $\overline{RAS}$ esté activa. Cada $\overline{CAS}$ sucesiva selecciona una columna de la fila especificada. De este modo, después de un ciclo en modo página rápido, todas las direcciones de la fila seleccionada se habrán leído o escrito, dependiendo del nivel de $R/\overline{W}$. Por ejemplo, un ciclo en modo página rápido para la DRAM de la Figura 12.17 requiere que la señal $\overline{CAS}$ se active 1024 veces para cada fila seleccionada mediante la señal $\overline{RAS}$.

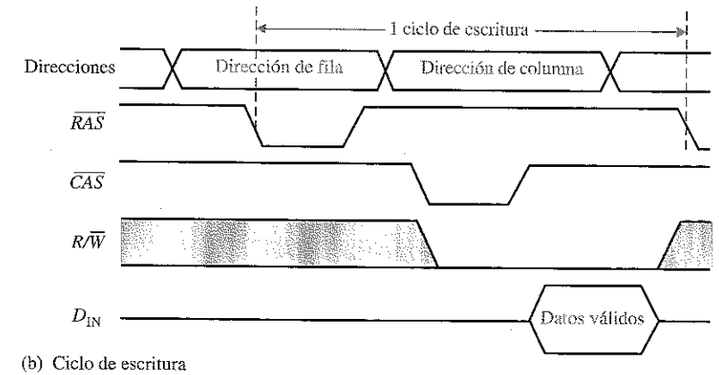
En la Figura 12.20 se presenta un diagrama de tiempos que ilustra el funcionamiento básico en modo página rápida para la operación de lectura. Cuando $\overline{CAS}$ pasa a su estado no activo (ALTO), desactiva las salidas de datos. Por tanto, la transición de la señal $\overline{CAS}$ al nivel ALTO sólo debe producirse después de que el sistema externo almacene los datos válidos en un latch.

Ciclos de refresco. Como ya sabemos, las DRAM se basan en el almacenamiento de carga en un condensador para cada bit de memoria de la matriz. Esta carga se degrada (se pierde) con el tiempo y la temperatura, por lo que cada bit se debe refrescar (recargar) periódicamente para mantener el estado correcto del bit. Típicamente, una DRAM se debe refrescar cada 8 ms o 16 ms, aunque en algunos dispositivos el periodo de refresco puede exceder 100 ms.

Una operación de lectura refresca automáticamente todas las direcciones de la fila seleccionada. Sin embargo, en aplicaciones típicas, no siempre se puede predecir cuán a menudo se producirá un ciclo de lectura y, por tanto, no se puede depender de que un ciclo de lectura se efectúe frecuentemente para evitar la pérdida de datos. En consecuencia, en los sistemas DRAM se deben implementar ciclos de refresco especiales.



(a) Ciclo de lectura



(b) Ciclo de escritura

FIGURA 12.19
Diagrama de tiempos para los ciclos de lectura y escritura normales.

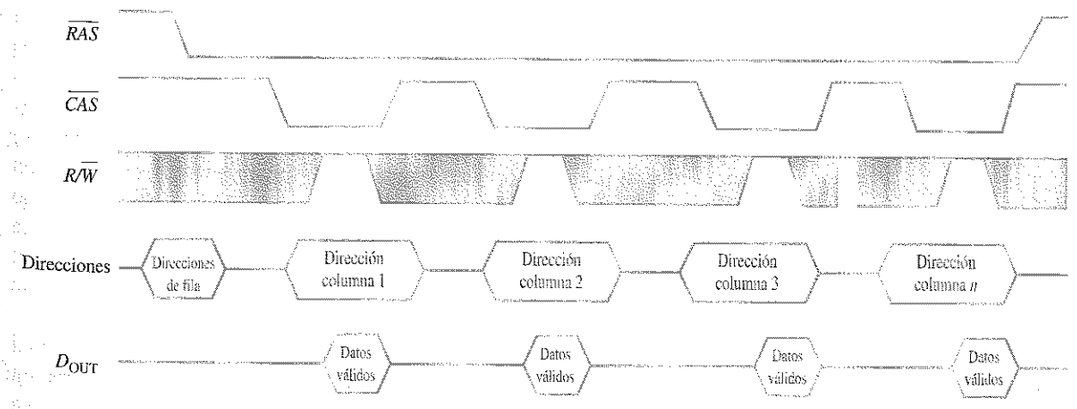


FIGURA 12.20
Cronograma del modo página rápido para la operación de lectura.

El *refresco a ráfagas* y el *refresco distribuido* son los dos modos básicos de las operaciones de refresco. En el refresco a ráfagas, todas las filas de la matriz de memoria se recargan consecutivamente en cada periodo de refresco. Para una memoria con un periodo de refresco de 8 ms, se produce una vez cada 8 ms un refresco a ráfagas de todas las filas.

En el refresco distribuido, cada fila se refresca en intervalos entremezclados con los ciclos de lectura y escritura normales. Por ejemplo, en la memoria de la Figura 12.17, que tiene 1024 filas, un periodo de refresco de 8 ms exige que se recargue una fila cada 8 ms/1024 = 7,8 µs, cuando se utiliza el refresco distribuido.

Los dos tipos de operaciones de refresco son: *refresco sólo* $\overline{RAS}$ (*$\overline{RAS}$ only refresh)* y *refresco* $\overline{CAS}$ *antes de* $\overline{RAS}$ (*$\overline{CAS}$ before $\overline{RAS}$)*. El refresco sólo $\overline{RAS}$ consiste en una transición de la señal $\overline{RAS}$ a nivel BAJO (estado activo), que almacena la dirección de la fila en el latch para realizar el refresco, mientras que la línea $\overline{CAS}$ permanece a nivel ALTO (estado inactivo) a lo largo del ciclo. Se utiliza un contador externo para proporcionar las direcciones de fila en este tipo de operación.

El refresco $\overline{CAS}$ antes de $\overline{RAS}$ se inicia cuando la línea $\overline{CAS}$ pasa a nivel BAJO y a continuación la línea $\overline{RAS}$ pasa a nivel BAJO. Esta secuencia activa un contador de refresco interno que genera la dirección de fila para la que se debe realizar la recarga. Esta dirección se conmuta mediante el selector de datos hacia el decodificador de filas.

Tipos de memoria DRAM

Ahora que hemos aprendido los conceptos básicos de una memoria DRAM, echemos un breve vistazo a los tipos principales existentes. Dichos tipos son la *DRAM con modo página rápido (Fast Page Mode, FPM DRAM)*, la *DRAM con salida de datos extendida (Extended Data Output, EDO DRAM)*, la *DRAM con salida de datos extendida en ráfaga (Burst Extended Data Output, BEDO DRAM)* y la *DRAM síncrona (Synchronous DRAM, SDRAM)*.

FPM DRAM. Ya hemos descrito anteriormente la operación en modo de página rápido. Este tipo de DRAM ha sido tradicionalmente el más común y es el que se ha usado en las computadoras hasta el desarrollo de la EDO DRAM. Recuerde que una página de la memoria está formada por todas las direcciones de columna contenidas en una misma dirección de fila.

Como ya hemos visto, la idea básica de la **FPM DRAM** se basa en la probabilidad de que las siguientes direcciones de memoria a las que haya que acceder se encuentren en la misma fila (en la misma página). Afortunadamente, esto sucede en un gran porcentaje de las veces. El modo FPM ahorra tiempo, con respecto al acceso puramente aleatorio, porque en el modo FPM la dirección de fila se especifica una única vez para acceder a varias direcciones de columna sucesivas, mientras que en el acceso aleatorio puro, hay que especificar una dirección de fila para cada dirección de columna.

Recuerde que en una operación de lectura en modo de página rápido, la señal $\overline{CAS}$ tiene que esperar hasta que los datos válidos correspondientes a una dirección dada sean aceptados (enclavados) por el sistema externo (CPU), antes de volver a su estado inactivo. Cuando $\overline{CAS}$ pasa a su estado inactivo, se desactivan las salidas de datos. Esto significa que la siguiente dirección de columna no puede ser generada

hasta que los datos correspondientes a la dirección de columna actual sean transferidos a la CPU. Esto limita la velocidad de acceso a las columnas situadas dentro de una página.

EDO DRAM. La memoria DRAM con salida de datos extendida, algunas veces denominada *DRAM con modo hiperpágina*, es muy similar a la FPM DRAM. La diferencia fundamental es que la señal $\overline{CAS}$ en la **EDO DRAM** no desactiva los datos de salida cuando pasa a su estado de inactividad, porque se pueden mantener los datos válidos correspondientes a la dirección actual hasta que $\overline{CAS}$ vuelva a activarse. Esto significa que se puede acceder a la siguiente dirección de columna antes de que el sistema externo acepte los datos válidos actuales. La idea es acelerar el tiempo de acceso.

BEDO DRAM. La DRAM con salida de datos extendida en ráfaga es una EDO DRAM con la capacidad de generar ráfagas de direcciones. Recuerde, de nuestra explicación sobre la SRAM síncrona de ráfaga, que la función de ráfaga de direcciones permite generar internamente hasta cuatro direcciones a partir de una única dirección externa, lo que ahorra tiempo de acceso. Este mismo concepto se aplica a la **BEDO DRAM**.

SDRAM. Para poder estar a la altura de la siempre creciente velocidad de los microprocesadores, son necesarias memorias DRAM más rápidas. La DRAM síncrona es uno de los esfuerzos más recientes en este sentido. Al igual que la RAM estática síncrona explicada anteriormente, la operación de la memoria **SDRAM** está sincronizada con el reloj del sistema, con el que también opera el microprocesador de un sistema informático. Las mismas ideas básicas descritas en relación con la SRAM síncrona de ráfaga se pueden aplicar a la memoria SDRAM.

Esta operación de tipo síncrono hace que la memoria SDRAM sea totalmente diferente de los otros tipos de DRAM asíncrona previamente mencionados. Con las memorias asíncronas, el microprocesador se ve obligado a esperar a que la DRAM complete sus operaciones internas. Con la operación de tipo síncrono, por el contrario, la DRAM enclava las direcciones, los datos y la información de control generados por el procesador, bajo control del reloj del sistema. Esto permite al procesador gestionar otras tareas mientras se están realizando las operaciones de lectura o escritura en memoria, en lugar de tener que esperar a que la memoria realice su tarea, como es el caso en los sistemas asíncronos.

REVISIÓN DE LA SECCIÓN 12.2

1. Enumerar dos tipos de SRAM.
2. ¿Qué es una memoria caché?
3. Explicar en qué se diferencian las SRAM y las DRAM.
4. Describir la operación de refresco de una DRAM.
5. Enumerar cuatro tipos de DRAM.

12.3 ■ MEMORIAS DE SÓLO LECTURA (ROM)

Como ya se ha mencionado anteriormente, una ROM mantiene de forma permanente o semipermanente los datos almacenados, que pueden ser leídos, pero, o no se pueden

cambiar en absoluto, o se requiere un equipo especial para ello. Una ROM almacena datos que se utilizan repetidamente en las aplicaciones, tales como tablas, conversiones o instrucciones programadas para la inicialización y el funcionamiento de un sistema. Las ROM mantienen los datos almacenados cuando se desconecta la alimentación.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Enumerar los tipos de ROM. ■ Describir una celda básica de almacenamiento ROM de máscara. ■ Explicar cómo se leen los datos de una ROM. ■ Estudiar la organización interna de una ROM típica. ■ Estudiar algunas aplicaciones de las ROM.

La familia de las memorias ROM

La Figura 12.21 nos muestra cómo se clasifican las memorias ROM semiconductoras. La ROM de máscara es un tipo de memoria en la que los datos se almacenan permanentemente en la memoria durante el proceso de fabricación. La **PROM**, o ROM programable, es aquel tipo de ROM en la que el usuario, con ayuda de equipos especializados, almacena eléctricamente los datos. Tanto la ROM de máscara como la PROM pueden ser de cualquier tecnología MOS o bipolar. La **EPROM**, o memoria PROM borrable (erasable PROM) es exclusivamente un dispositivo MOS. La **UV EPROM** puede ser programada eléctricamente por el usuario, pero los datos almacenados deben borrarse mediante la exposición a la luz ultravioleta durante un periodo de varios minutos. La PROM borrable eléctricamente (**EEPROM** o E^2PROM, Electrically Erasable PROM) se puede borrar en unos pocos milisegundos.

La ROM de máscara

Normalmente, la ROM de máscara se denomina simplemente ROM. Es una memoria programada de forma permanente durante el proceso de fabricación, para proporcionar funciones estándar de uso extendido, tales como conversiones populares, o para proporcionar funciones especificadas por el usuario. Una vez que se programa la memoria, ésta no puede cambiarse. La mayoría de los circuitos integrados ROM utilizan la presencia o ausencia de una conexión de transistor en una unión fila/columna para representar un 1 o un 0.

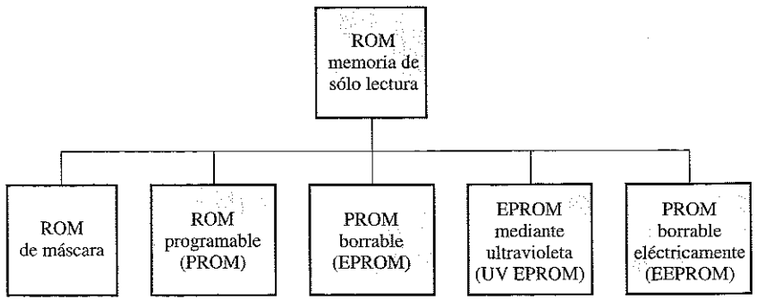


FIGURA 12.21
La familia de memorias ROM.

La Figura 12.22 muestra celdas MOS de una ROM. La presencia de una conexión desde una línea de fila a la puerta de un transistor representa un 1 en esa posición, ya que, cuando la línea de fila está a nivel ALTO, todos los transistores con conexión de puerta a esa línea de fila conducen, y ponen a nivel ALTO (1) a las líneas de columna asociadas. En las uniones de fila/columna en las que no existe conexión de puerta, las líneas de columna permanecen a nivel BAJO (0) cuando se direcciona la fila.

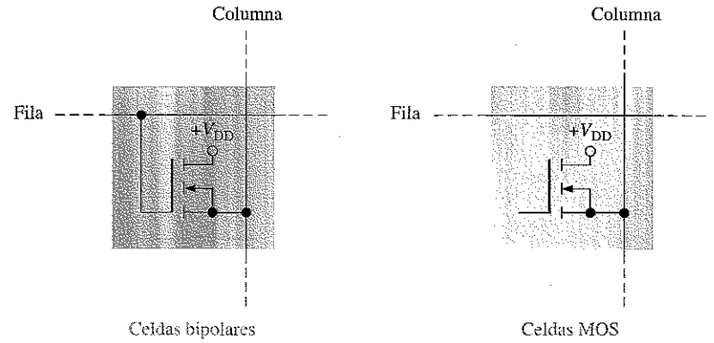


FIGURA 12.22
Celdas ROM.

Una ROM básica

Para ilustrar el concepto de ROM, la Figura 12.23 muestra una matriz ROM simplificada. Los cuadrados en color gris claro representan los 1s almacenados, y los cuadrados en gris oscuro representan los 0s almacenados. El procedimiento de lectura básico es el siguiente: cuando se aplica un código de dirección binario a las líneas de entrada de dirección, la línea de la fila correspondiente se pone a nivel ALTO. Este nivel ALTO se conecta a las líneas de las columnas a través de los transistores en cada unión (celda) donde se almacena un 1. En cada celda en la que se almacena un 0, la línea de columna permanece a nivel BAJO, debido a la resistencia de terminación. Las líneas de las columnas constituyen la salida de datos. Los ocho bits de datos almacenados en la fila seleccionada se presentan en las líneas de salida.

Como puede verse, la ROM de ejemplo de la Figura 12.23 está organizada en 16 direcciones, cada una de las cuales almacena 8 bits de datos. Por tanto, se trata de una ROM de 16 x 8 (16 por 8) y su capacidad total es de 128 bits, o 16 bytes.

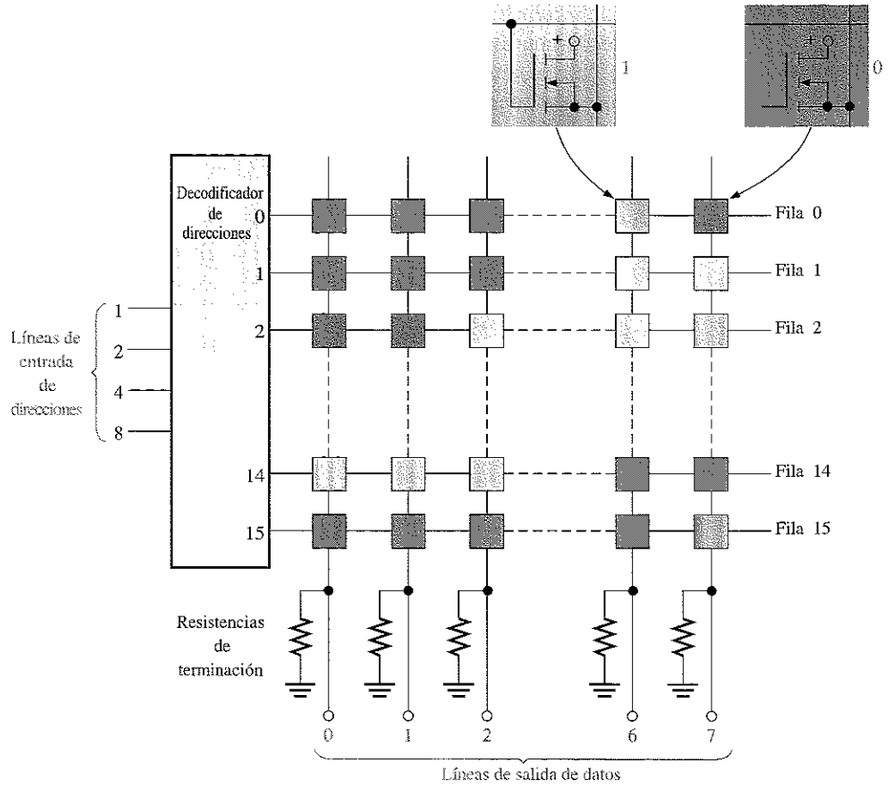


FIGURA 12.23
Matriz ROM de 16 × 8 bits.

EJEMPLO 12.1 Dibujar una ROM similar a la de la Figura 12.23, programada para convertir a código Gray números binarios de 4 bits.

Solución. Repase, en el Capítulo 2, el código Gray. Se ha desarrollado la Tabla 12.1 con el fin de utilizarla para programar la ROM.

La matriz ROM resultante se muestra en la Figura 12.24. Se puede ver que un código binario en las líneas de dirección de entrada produce el correspondiente código Gray en las líneas de salida (columnas). Por ejemplo, cuando se aplica el número binario 0110 a las líneas de dirección de entrada, se selecciona la dirección 6, que almacena el código Gray 0101.

Problema relacionado. Utilizando la Figura 12.24, determinar el código Gray de salida cuando se aplica el código binario 1011 a las líneas de dirección de entrada.

TABLA 12.1

Binario				Gray			
B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

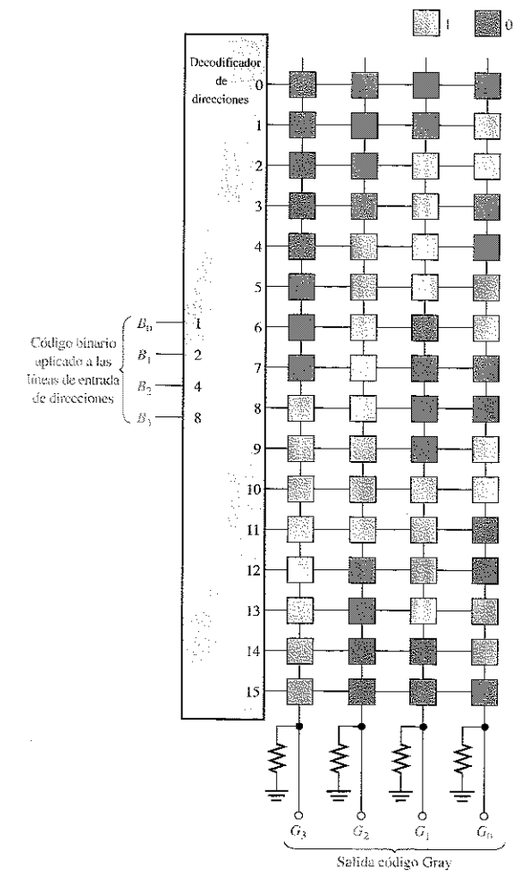


FIGURA 12.24

Representación de una ROM programada como convertidor de código binario - Gray.

Organización interna de la ROM

La mayoría de los circuitos integrados ROM tiene una organización interna algo más compleja que la de la ROM básica del ejemplo que acabamos de presentar. Para ilustrar cómo se estructura un CI ROM se utiliza un dispositivo de 1024 bits organizado en una matriz de 256 x 4. El símbolo lógico se muestra en la Figura 12.25. Cuando se aplica cualquiera de los 256 códigos binarios (ocho bits) a las líneas de dirección, se presentan cuatro bits de datos en las salidas si las entradas de habilitación del chip se encuentran a nivel BAJO (existen ocho líneas de direccionamiento, ya que $2^8 = 256$). Aunque la estructura de 256 x 4 de este dispositivo parece indicar que tiene 256 filas y 4 columnas en la matriz de memoria, éste no es el caso en realidad. Realmente la matriz de memoria es una matriz de 32 x 32 (32 filas y 32 columnas), como muestra el diagrama de bloques de la Figura 12.26

NOTAS INFORMÁTICAS
La memoria ROM se usa en las computadoras personales para almacenar lo que se denomina BIOS (Basic Input/Output Services, servicios básicos de entrada/salida). Se trata de programas que se emplean para llevar a cabo funciones fundamentales de soporte y supervisión en la computadora. Por ejemplo, los programas de BIOS almacenados en la ROM controlan determinadas funciones de vídeo, proporcionan la función de formateo de discos, exploran el teclado para detectar las pulsaciones y controlan ciertas funciones de impresión.

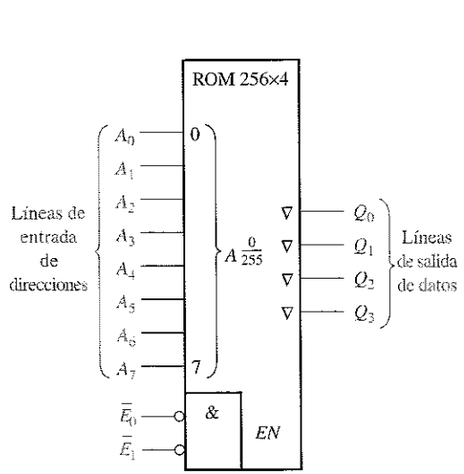
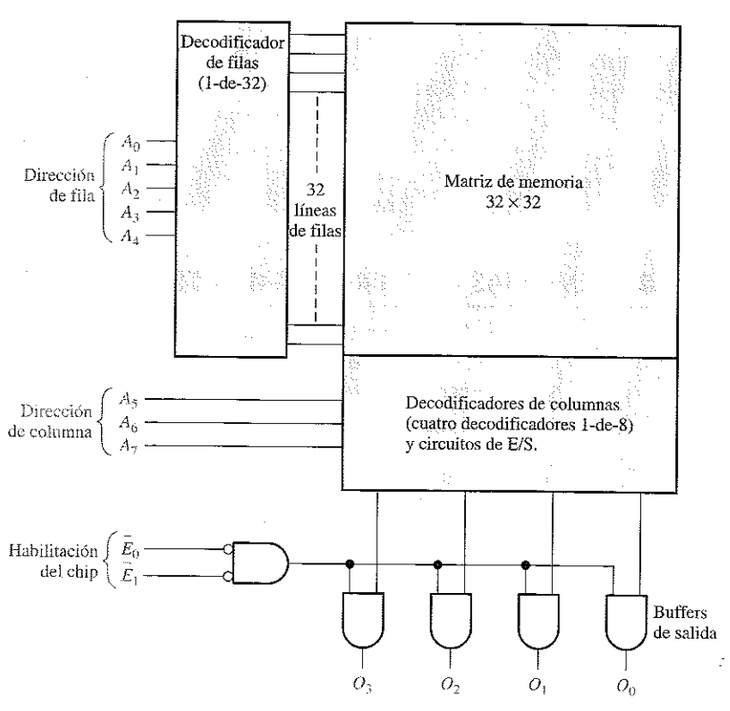


FIGURA 12.25
Símbolo lógico de una ROM de 256 × 4. El identificativo $A \frac{0}{255}$ significa que el código de dirección de 8 bits selecciona direcciones de 0 a 255.

La ROM de la Figura 12.26 funciona de la manera siguiente: cinco de las ocho líneas de dirección (A_0 hasta A_4) se decodifican mediante el decodificador de filas (comúnmente denominado decodificador *Y*) para seleccionar una de las 32 filas. Tres de las ocho líneas de dirección (A_5 hasta A_7) se decodifican mediante el decodificador de columnas (denominado comúnmente decodificador *X*) para seleccionar cuatro de las 32 columnas. En realidad, el decodificador de columnas está formado por cuatro decodificadores 1-de-8 (selectores de datos), como se muestra en la Figura 12.26.

El resultado de esta estructura es que, al aplicar un código de dirección de 8 bits (A_0 hasta A_7), aparece una palabra de datos de 4 bits en las salidas de datos. Las líneas de habilitación del chip ($\bar{E}_0$ y $\bar{E}_1$) deben estar a nivel bajo para activar los buffers de salida. Este tipo de organización interna (arquitectura) es típica de diversos circuitos integrados ROM, de distintas capacidades.

FIGURA 12.26
ROM típica. Este ejemplo en particular es una ROM de 1024 bits con una organización de 256 × 4 basada en una matriz de 32 × 32.



Tiempo de acceso de la ROM

En la Figura 12.27 se presenta un diagrama de tiempos típico que ilustra el tiempo de acceso a la ROM. El **tiempo de acceso** de una ROM, t_a, es el tiempo que transcurre desde que se aplica un código de dirección válido en las líneas de entrada hasta que aparecen los datos válidos en las líneas de salida. El tiempo de acceso se puede también medir desde que se activa la entrada de habilitación del chip ($\bar{E}$) hasta que aparecen los datos válidos en la salida, cuando ya se encuentra una dirección válida en las líneas de entrada.

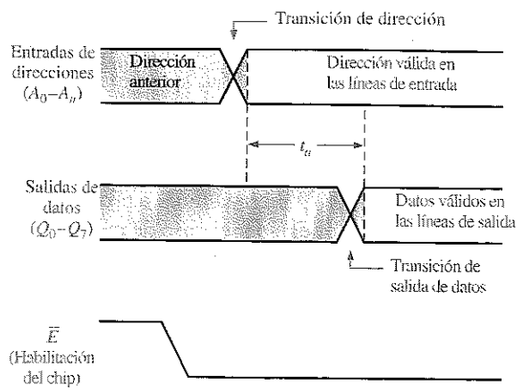


FIGURA 12.27
Tiempo de acceso de una ROM (t_a) desde el cambio de dirección hasta la salida de datos, con la entrada de habilitación del chip activa.

REVISIÓN DE LA SECCIÓN 12.3

1. ¿Cuál es la capacidad de almacenamiento en bits de una ROM con una estructura de 512×8?
2. Explicar la finalidad de las salidas de tres estados.
3. ¿Cuántos bits de dirección son necesarios en una memoria de 2048 bits organizada como una memoria de 256×8?

12.4 ■ MEMORIAS ROM PROGRAMABLES (PROM Y EPROM)

Las PROM son básicamente iguales que las ROM de máscara, una vez que han sido programadas. La diferencia consiste en que las PROM salen de fábrica sin estar programadas y se programan a medida para satisfacer las necesidades del usuario.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Distinguir entre una ROM de máscara y una PROM. ■ Describir una celda básica de almacenamiento de una PROM. ■ Utilizar las EPROM, incluyendo las UV EPROM y EEPROM. ■ Analizar un ciclo de programación de una PROM.

Memorias PROM

Las PROM utilizan algún tipo de mecanismo de fundición para almacenar bits, donde un *hilo* de memoria se funde o queda intacto para representar un 0 o un 1. El proceso de fundición es irreversible; una vez que una PROM ha sido programada no puede cambiarse.

La Figura 12.28 muestra una matriz MOS de una PROM con hilos fusibles. Los fusibles se introducen en la PROM (durante el proceso de fabricación) entre la fuente del transistor de cada celda y su línea de columna. Durante el proceso de programación, se introduce una corriente adecuada a través del hilo fusible para fundirlo y que permanezca abierto, almacenando de esta manera un 0. El fusible se deja intacto para almacenar un 1.

Los tres tipos básicos de tecnologías de fusibles utilizados en las PROM son las conexiones de metal, las conexiones de silicio y las uniones *pn*. A continuación, se da una breve descripción de cada una de ellas.

1. Las conexiones de metal se realizan con materiales como el nicromo. Cada bit de la matriz de memoria se representa mediante una conexión separada. Durante la programación, la conexión puede fundirse o quedar intacta. Básicamente, esto se realiza direccionando primero una determinada celda, y luego aplicando una cantidad de corriente suficientemente alta como para hacer que la conexión se abra.
2. Las conexiones de silicio están constituidas por tiras estrechas y alargadas de silicio policristalino. La programación de estos fusibles requiere que las conexiones se fundan por el paso de una cantidad de corriente adecuada a su través. Esta cantidad de corriente hace que aumente la temperatura en el fusible, lo que origina que se oxide el silicio, formando un aislante alrededor de la conexión que ahora está abierta.
3. La tecnología de uniones cortocircuitadas, o de migración inducida por avalancha, consiste básicamente en dos uniones *pn* dispuestas una frente a la otra. Durante el proceso de programación, una de las uniones de los diodos entra en avalancha, y el voltaje y el calor resultantes hacen que los iones de aluminio migren y cortocircuiten la unión. La unión restante se utiliza posteriormente como diodo polarizado en directa para representar un bit de datos.

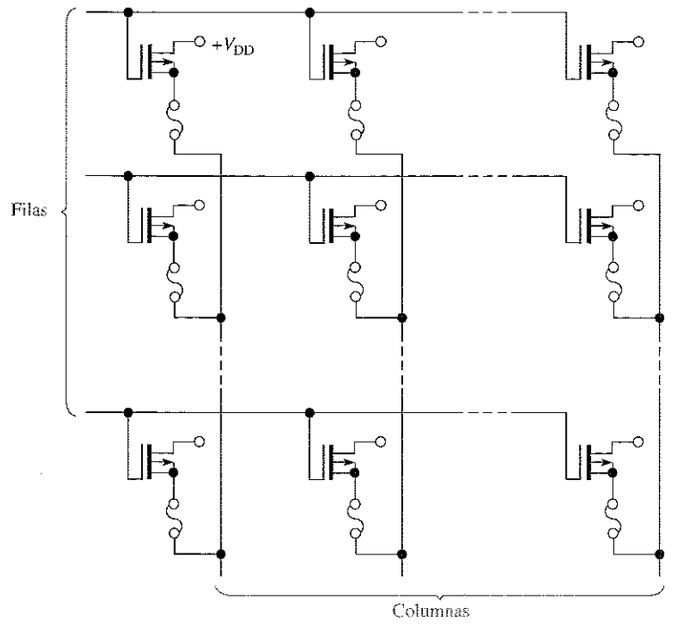


FIGURA 12.28
Matriz PROM MOS con hilos fusibles. Generalmente, todos los drenadores se conectan a V_{DD}.

Programación de memorias PROM

Normalmente, una PROM se programa insertándola en un dispositivo especial denominado *programador de PROM*. Esencialmente, la programación se realiza mediante la configuración simplificada de la Figura 12.29. Se selecciona una dirección mediante la configuración de interruptores de las líneas de dirección, y luego se aplica un impulso a aquellas líneas de salida correspondientes a las posiciones de los bits donde se tienen que almacenar los ceros (inicialmente, las PROM tienen 1s en todas sus direcciones). Estos impulsos funden los fusibles, creando el patrón de bits requerido. Después, se selecciona la siguiente dirección y se repite el proceso. Esta secuencia se realiza automáticamente mediante un programador de PROM controlado por software.

Memorias EPROM

Una EPROM es una PROM borrable. A diferencia de una PROM ordinaria, una EPROM puede ser reprogramada si antes se borra el programa existente en la matriz de memoria.

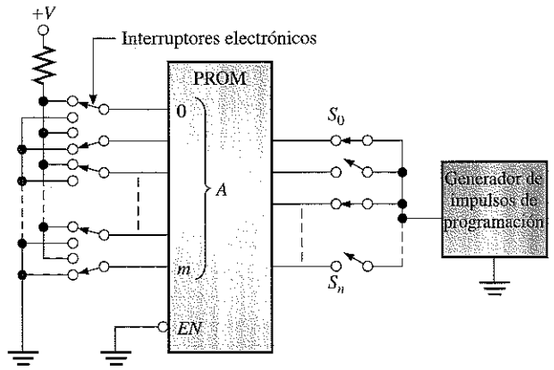


FIGURA 12.29
Configuración simplificada para la programación de una PROM.

Una EPROM utiliza una matriz NMOSFET con una estructura de puerta aislada. La puerta del transistor aislada no tiene ninguna conexión eléctrica y puede almacenar una carga eléctrica durante un periodo de tiempo indefinido. Los bits de datos en este tipo de matriz se representan mediante la presencia o ausencia de una carga almacenada en la puerta. El borrado de un bit de datos es un proceso que elimina la carga de la puerta.

Los dos tipos fundamentales de memorias PROM borrables son las PROM borrables por rayos ultravioleta (UV EPROM) y las PROM borrables eléctricamente (EEPROM).

UV EPROM. Una UV EPROM se puede reconocer por la ventana de cuarzo transparente de su encapsulado, como se muestra en la Figura 12.30. La puerta aislada del **FET** de una EPROM ultravioleta está "flotando" dentro de un material óxido aislante. El proceso de programación hace que los electrones sean eliminados de la puerta flotante. El borrado se realiza mediante la exposición del chip de la matriz de memoria a una radiación ultravioleta de alta intensidad, a través de la ventana de cuarzo en la parte superior del encapsulado. La carga positiva almacenada en la puerta se neutraliza después de un periodo de tiempo de entre unos minutos y una hora de exposición.

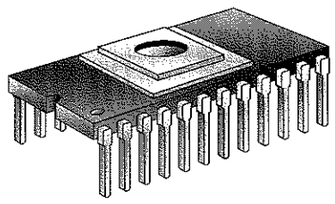


FIGURA 12.30
Encapsulado de una PROM borrable por rayos ultravioleta.

EEPROM. Las PROM borrables eléctricamente se pueden borrar y programar mediante impulsos eléctricos. Ya que se pueden grabar y borrar eléctricamente, las EEPROM se pueden reprogramar dentro del propio circuito final, lo que permite reconfigurar cualquier sistema.

Los dos tipos de EEPROM son la MOS de puerta flotante y la de silicio óxido nitroso metal (MNOS, metal nitride-oxide silicon). La aplicación de una tensión en la puerta de control de la estructura de puerta flotante permite la eliminación y el almacenamiento de la carga en la puerta flotante.

Un ejemplo de EPROM

El circuito integrado 27C16B es un ejemplo de EPROM. Su funcionamiento es representativo de otras memorias EPROM típicas, de distintos tamaños. Como muestra el símbolo lógico de la Figura 12.31, este dispositivo tiene 2048 direcciones (2^{11} = 2048), cada una con ocho bits. Obsérvese que las ocho salidas son triestado (∇).

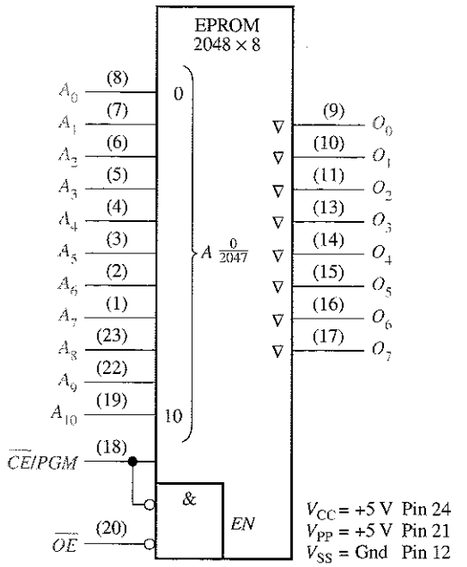


FIGURA 12.31
Símbolo lógico de una EPROM 2048 × 8.

Para leer la memoria, la entrada de habilitación de salida ($\overline{OE}$) tiene que estar a nivel BAJO y la entrada de habilitación de chip/programación ($\overline{CE}/PGM$, chip enable/program) también a nivel BAJO. Para borrar los datos almacenados, el dispositivo se expone a una luz ultravioleta de alta intensidad que pasa a través de la tapa transparente. Una típica lámpara UV sin filtro de 12 mW/cm^2 borrará los datos en unos 20 o 25 minutos. Al igual que en la mayoría de las EPROM, tras el borrado todos los bits son 1s. La luz ambiente normal contiene la longitud de onda ultravioleta adecuada

para hacer que se realice el borrado tras un periodo de tiempo suficiente. Por tanto, la ventana transparente se ha de mantener tapada.

Para programar el dispositivo, se aplica una tensión continua alta a V_{PP} (que normalmente es +5 V o +3 V), estando $\overline{OE}$ a nivel ALTO. Los ocho bits de datos que van a ser programados en una dirección determinada se aplican a las salidas (O_0 hasta O_7) y se selecciona la dirección en las entradas A_0 hasta A_7. A continuación, se aplica un impulso a nivel ALTO de entre 10 ms y 55 ms a la entrada $\overline{CE}/PGM$. Las direcciones se pueden programar en cualquier orden.

En la Figura 12.32 se muestra un cronograma para el modo de programación. Normalmente, estas señales las produce un programador de dispositivos EPROM.

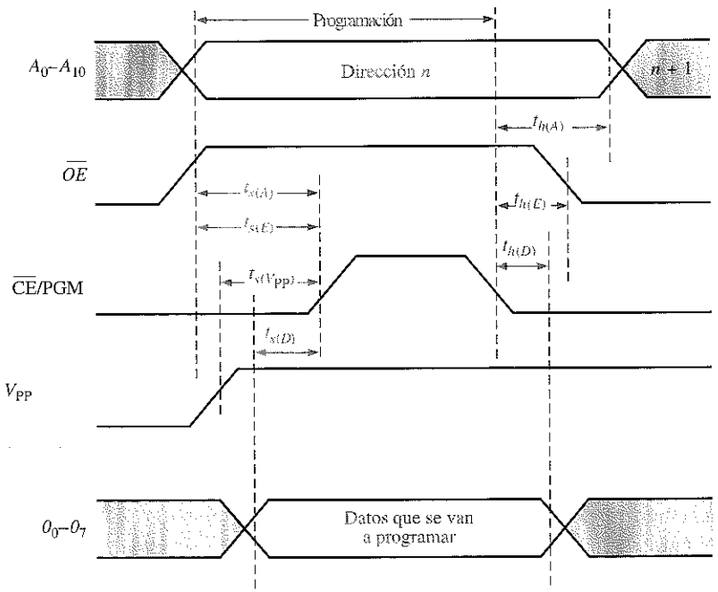


FIGURA 12.32
Diagrama de tiempos de un ciclo de programación de una EPROM 2048 × 8, que indica los tiempos de establecimiento (t_s) y de mantenimiento (t_h) críticos.

REVISIÓN DE LA SECCIÓN 12.4

1. ¿En qué se diferencian las memorias PROM de las ROM?
2. Después del borrado, ¿todos los bits de una EPROM típica son 1s o 0s?
3. ¿Cuál es el modo normal de funcionamiento de una PROM?

12.5 ■ MEMORIAS FLASH

La memoria ideal debería tener una alta capacidad de almacenamiento, ser no volátil, disponer de capacidad de lectura y escritura en el propio sistema, tener una velocidad de operación comparativamente rápida y ser efectiva en términos de coste. Las tecnologías de memorias tradicionales, como ROM, PROM, EPROM, EEPROM, SRAM y DRAM exhiben, cada una de ellas, una o más de estas características, pero ninguna tecnología las tiene todas, salvo la memoria flash.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar las características básicas de una memoria flash. ■ Describir la operación básica de una célula de memoria flash. ■ Comparar las memorias flash con otros tipos de memoria.

Las **memorias flash** son memorias de lectura/escritura de alta densidad (alta densidad equivale a gran capacidad de almacenamiento de bits) no volátiles, lo que significa que pueden almacenarse los datos indefinidamente en ausencia de alimentación. Estas memorias se utilizan frecuentemente en lugar de las unidades de disquete o de las unidades de disco duro de baja capacidad en las computadoras portátiles.

La característica de alta densidad significa que puede incluirse un gran número de celdas en un área de superficie dada del chip; es decir, cuanto más alta sea la densidad, más bits podrán almacenarse en un chip de un tamaño determinado. Esta alta densidad se consigue en las memorias flash con una célula de almacenamiento compuesta por un único transistor MOS de puerta flotante. El bit de datos se almacena como una carga o una ausencia de carga en la puerta flotante, dependiendo de si se desea almacenar un 0 o un 1.

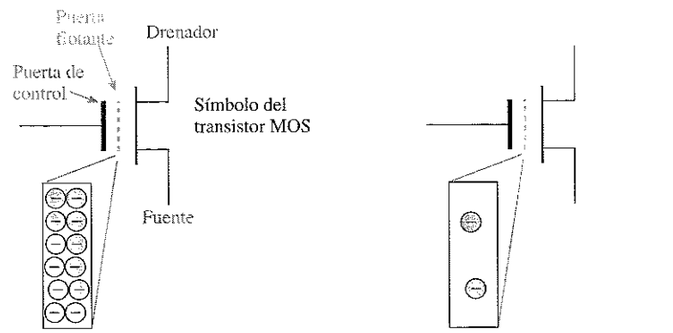
Célula de memoria flash

La Figura 12.33 representa una célula monotransistor de una memoria flash. El transistor MOS de puerta apilada consta de una puerta de control y una puerta flotante, además del drenador y la fuente. La puerta flotante almacena electrones (carga) si se aplica la suficiente tensión a la puerta de control. *Se almacena un 0 cuando existe una cantidad significativa de carga y un 1 cuando la carga es menor, o inexistente*. La cantidad de carga presente en la puerta flotante determina si el transistor se activará y conducirá corriente del drenador a la fuente cuando se aplique una tensión de control durante una operación de lectura.

Operación básica de la memoria flash

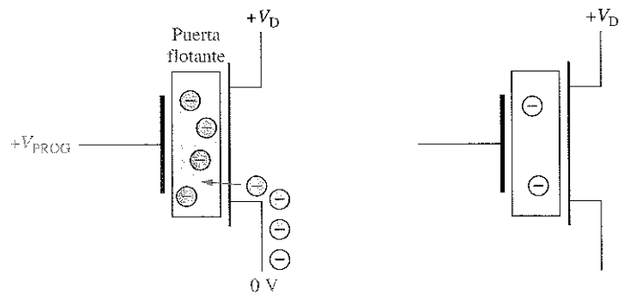
Hay tres operaciones principales en una memoria flash: la operación de *programación*, la operación de *lectura* y la operación de *borrado*.

Programación. Inicialmente, todas las células se encuentran en el estado 1, porque la carga fue eliminada de las células en una operación previa de borrado. La operación de programación añade electrones (carga) a la puerta flotante de aquellas células que deban almacenar un 0. No se añade carga a aquellas células que deban almacenar un 1. La aplicación a la puerta de control de una tensión suficientemente positiva con respecto a la fuente, durante la programación, atrae electrones a la puerta flotante, como indica la Figura 12.34. Una vez programada, una célula puede retener la carga durante 100 años sin necesidad de aplicar una alimentación externa.



Muchos electrones = más carga = 0 almacenado Pocos electrones = menos carga = 1 almacenado

FIGURA 12.33
La célula de almacenamiento de una memoria flash.



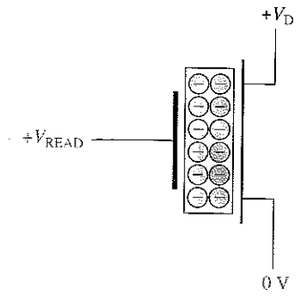
Para almacenar un 0, se aplica a la puerta de control una tensión suficientemente positiva con respecto a la fuente, para añadir carga a la puerta flotante durante la programación.

Para almacenar un 1, no se añade ninguna carga, dejándose la célula en el estado borrado.

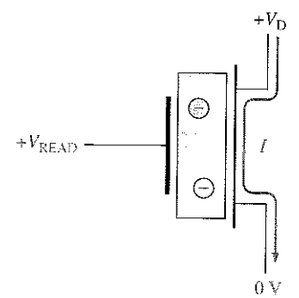
FIGURA 12.34
Ilustración simplificada del proceso de almacenamiento de un 0 o un 1 en una célula flash durante la operación de programación.

Lectura. Durante una operación de lectura, se aplica una tensión positiva a la puerta de control. La cantidad de carga presente en la puerta flotante de una célula determina si la tensión aplicada a la puerta de control activará, o no, el transistor. Si hay almacenado un 1, la tensión de la puerta de control es suficiente para activar el transistor. Si hay almacenado un 0, el transistor no se activará, porque la tensión de la puerta de control no es suficiente para contrarrestar la carga negativa almacenada en la puerta flotante. Piense en la carga de la puerta flotante como en una fuente de tensión que se opone a la tensión aplicada a la puerta de control durante la lectura. Desde este punto de vista, la carga de la puerta flotante asociada a un 0 almacenado evita que la tensión de la puerta de control alcance el umbral de activación, mientras que la carga pequeña o nula asociada con un 1 almacenado permite a la tensión de la puerta de control exceder dicho umbral de activación.

Cuando el transistor se activa, existe corriente desde el drenador hacia la fuente del transistor de la célula. La presencia o ausencia de esta corriente es detectada para indicar un 1 o un 0, respectivamente. Esta idea básica se ilustra en la Figura 12.35.



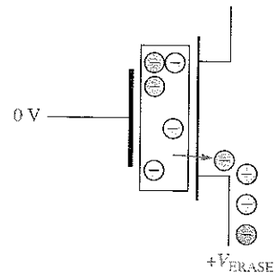
Cuando se lee un 0, el transistor permanece desactivado, porque la carga de la puerta flotante impide a la tensión de lectura exceder el umbral de activación.



Cuando se lee un 1, el transistor se activa, porque la ausencia de carga en la puerta flotante permite que la tensión de lectura exceda el umbral de activación.

FIGURA 12.35
La operación de lectura de una célula flash de una matriz.

Borrado. Durante una operación de borrado, se elimina la carga de todas las células de memoria. Para ello, se aplica a la fuente del transistor una tensión suficientemente positiva con respecto a la puerta de control. Esta polaridad es precisamente la opuesta a la utilizada durante la programación. Esta tensión atrae a los electrones de la puerta flotante y hace que ésta se vacíe de carga, como se ilustra en la Figura 12.36. Las memorias flash siempre se borran antes de volver a ser programadas.



Para borrar una célula, se aplica a la fuente una tensión suficientemente positiva con respecto a la puerta de control, con el fin de extraer la carga de la puerta flotante durante la operación de borrado.

FIGURA 12.36
Ilustración simplificada del proceso de eliminación de la carga de una célula durante la operación de borrado.

Matriz básica de una memoria flash

La Figura 12.37 muestra una matriz simplificada de células de memoria flash. Sólo se accede a una fila cada vez. Cuando una celda de una línea de bit dada se activa (un 1 almacenado) durante una operación de lectura, existirá corriente a través de la línea de bit, lo que producirá una caída de tensión a través de la carga activa. Esta caída de tensión se compara con una tensión de referencia mediante un circuito comparador, generándose un nivel de salida que indica que hay un 1. Si hay un 0 almacenado, no hay corriente en la línea de bit o ésta es muy pequeña, generándose un nivel opuesto a la salida del comparador.

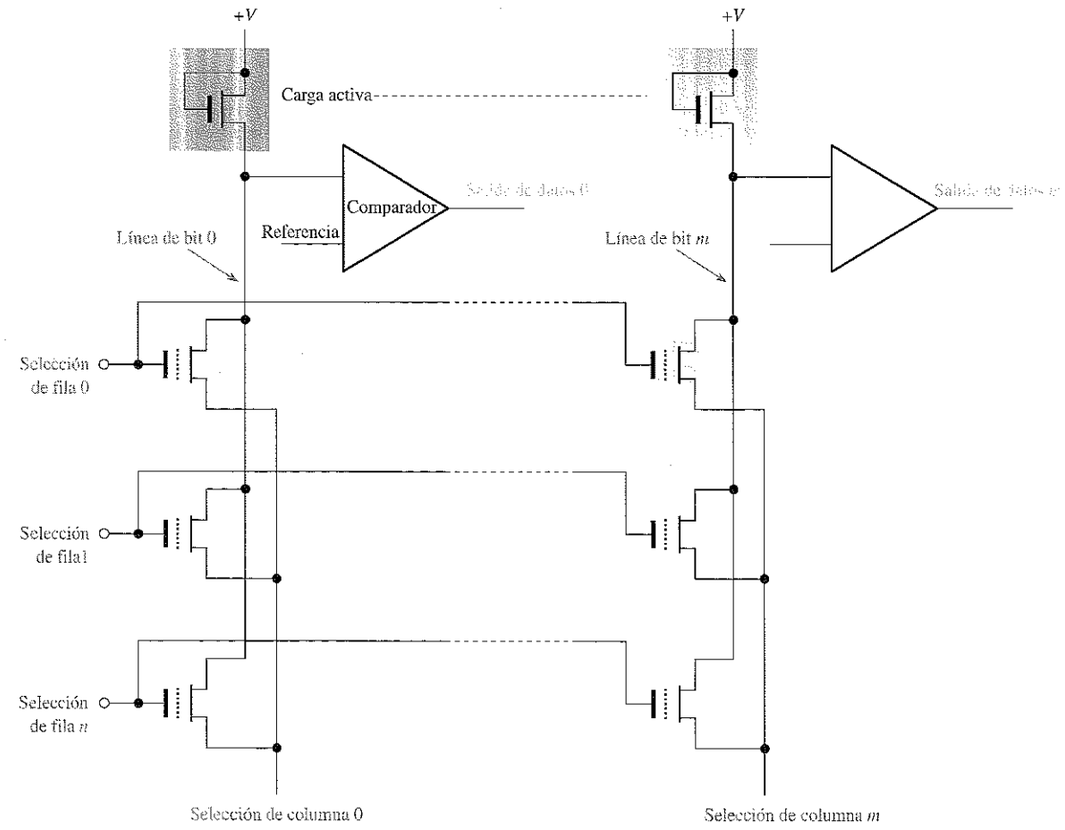


FIGURA 12.37
Matriz básica de una memoria flash.

Comparación de las memorias flash con otras memorias

Vamos a comparar las memorias flash con los otros tipos de memorias con los que ya estamos familiarizados.

Flash frente a ROM, EPROM y EEPROM. Las memorias de sólo lectura son dispositivos de alta densidad y no volátiles. Sin embargo, una vez que se ha programado una ROM, su contenido no puede nunca alterarse. También la programación inicial es un proceso costoso y que consume tiempo.

Aunque la EPROM es una memoria de alta densidad no volátil, sólo se puede borrar extrayéndola del sistema y utilizando luz ultravioleta. Sólo se puede reprogramar utilizando un equipo especial.

La EEPROM tiene una estructura de celda más compleja que la ROM y la EPROM y su densidad no es tan alta, aunque puede reprogramarse sin sacarse del sistema. Debido a su densidad mucho menor, el coste por bit es mayor que en las ROM y EPROM.

Una memoria flash se puede reprogramar fácilmente dentro del sistema ya que, esencialmente, es un dispositivo de LECTURA/ESCRITURA. La densidad de una memoria flash es comparable a la de la ROM y la EPROM, ya que ambas utilizan celdas de un único transistor. Una memoria flash (al igual que una ROM, EPROM o EEPROM) es no volátil, lo que permite almacenar los datos indefinidamente sin alimentación.

Flash frente a SRAM. Como se ha explicado, las memorias estáticas de acceso aleatorio son dispositivos de LECTURA/ESCRITURA volátiles. Una SRAM requiere una alimentación constante para mantener los datos almacenados. En muchas aplicaciones, se utiliza una batería de reserva para evitar la pérdida de datos, si la fuente de alimentación principal se apaga. Sin embargo, puesto que siempre existe la posibilidad de que la batería falle, el mantenimiento indefinido de los datos en una SRAM no se puede garantizar. Como las celdas de memoria de una SRAM son, básicamente, un latch formado por varios transistores, la densidad es relativamente baja.

Una memoria flash también es una memoria de LECTURA/ESCRITURA, pero, a diferencia de la SRAM, no es volátil. También, una memoria flash tiene una densidad mucho más alta que una SRAM.

Flash frente a DRAM. Las memorias de acceso aleatorio son dispositivos de LECTURA/ESCRITURA volátiles de alta densidad. Las DRAM no sólo requieren una alimentación constante para mantener los datos, sino que también los datos almacenados deben refrescarse frecuentemente. En muchas aplicaciones, debe utilizarse con la DRAM un almacenamiento de reserva, como por ejemplo un disco duro.

Las memorias flash tienen densidades más altas que las DRAM, porque una celda de memoria está formada por un transistor y no es necesario realizar el refresco, mientras que una celda DRAM es un transistor más un condensador que tiene que recargarse. Típicamente, una memoria flash consume menos potencia que una DRAM equivalente, y en muchas aplicaciones puede usarse para reemplazar un disco duro.

La Tabla 12.2 facilita un resumen de la comparación de las tecnologías de memorias.

TABLA 12.2
Comparación de los tipos de memoria.

Tipo de memoria	Volátil	Alta densidad	Celda con un sólo transistor	Reescribible en el sistema final
Flash	No	Sí	Sí	Sí
SRAM	Sí	No	No	Sí
DRAM	Sí	Sí	Sí	Sí
ROM	No	Sí	Sí	No
EPROM	No	Sí	Sí	No
EEPROM	No	No	No	Sí

REVISIÓN DE LA SECCIÓN 12.5

1. ¿Cuáles son los tipos de memoria no volátiles?
2. ¿Cuál es la ventaja principal de una memoria flash sobre una SRAM o una DRAM?
3. Enumerar los tres modos de operación de una memoria flash.

12.6 ■ EXPANSIÓN DE MEMORIAS

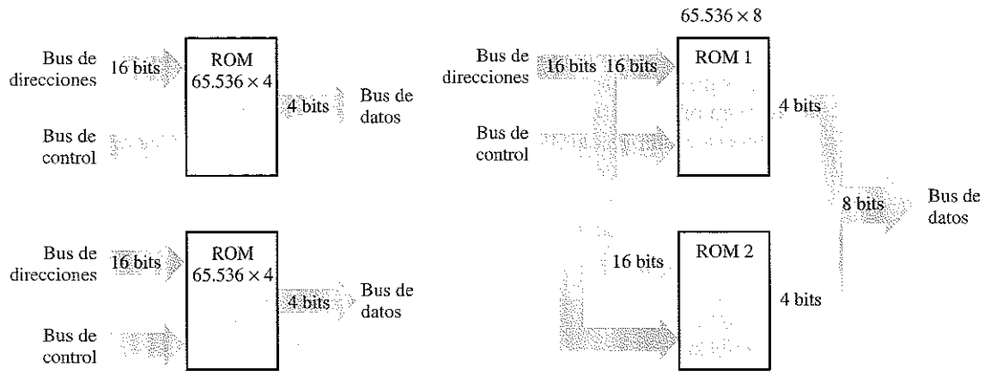
Las memorias disponibles se pueden ampliar para incrementar la longitud de palabra (número de bits en cada dirección) o la capacidad de palabra (número de direcciones diferentes), o ambas. La expansión de memoria se consigue añadiendo el número apropiado de chips de memoria a los buses de dirección, datos y control, como se explica en esta sección.

Al finalizar este capítulo, el lector deberá ser capaz de:

■ Definir la *expansión de la longitud de palabra.* ■ Mostrar cómo se amplía la longitud de palabra de una memoria. ■ Definir la *expansión de la capacidad de palabra.* ■ Mostrar cómo se amplía la capacidad de palabra de una memoria.

Expansión de la longitud de palabra

Para aumentar la **longitud de palabra** de una memoria, el número de bits del bus de datos debe aumentarse. Por ejemplo, se puede conseguir una longitud de palabra de 8 bits utilizando dos memorias de 4 bits, como se ilustra en la Figura 12.38(a). Como puede verse en la parte (b), el bus de direcciones de 16 bits se conecta normalmente a ambas memorias, de modo que la memoria combinada tenga el mismo número de direcciones (2^{16} = 65.536) que cada memoria individual. Los buses de datos de 4 bits de las dos memorias se combinan para formar un bus de datos de 8 bits. De este modo, cuando se selecciona una dirección, se producen ocho bits en el bus de datos, cuatro para cada memoria.



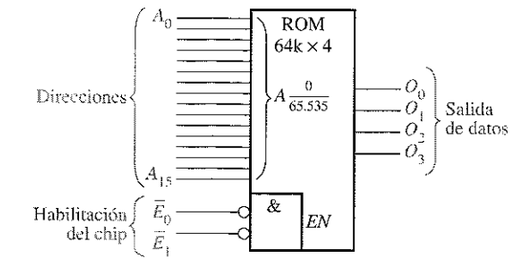
(a) Dos ROM de 65.536 x 4 separadas (b) Una ROM de 65.536 x 8 a partir de 2 ROM de 65.536 x 4

FIGURA 12.38
Expansión de dos ROM de 65.536 × 4 a una ROM de 65.536 × 8, que ilustra la expansión de la longitud de palabra.

El siguiente ejemplo muestra los detalles de la expansión de 65.536 x 4 a 65.536 x 8.

EJEMPLO 12.2 Expandir la ROM de 65.536 x 4 (64k x 4) de la Figura 12.39 para obtener una ROM de 64k x 8. Obsérvese que "64k" es una abreviatura aceptada para designar 65.536. Pero, ¿por qué no se emplea "65k"? Posiblemente, porque 64 es una potencia de dos.

FIGURA 12.39
Una ROM de 64k × 4.



Solución. Las dos ROM de 64k x 4 se conectan como se muestra en la Figura 12.40. Obsérvese que se accede a una dirección específica en la ROM1 y en la ROM2 a la vez. Los cuatro bits de la dirección seleccionada de la ROM1 y los cuatro bits correspondientes de la ROM2 se llevan en paralelo para formar una palabra de 8 bits en el bus de datos. Obsérvese también que un nivel BAJO en la línea de habilitación de chip $\overline{E}$, que actúa como un bus de control simple, activa *ambas* memorias.

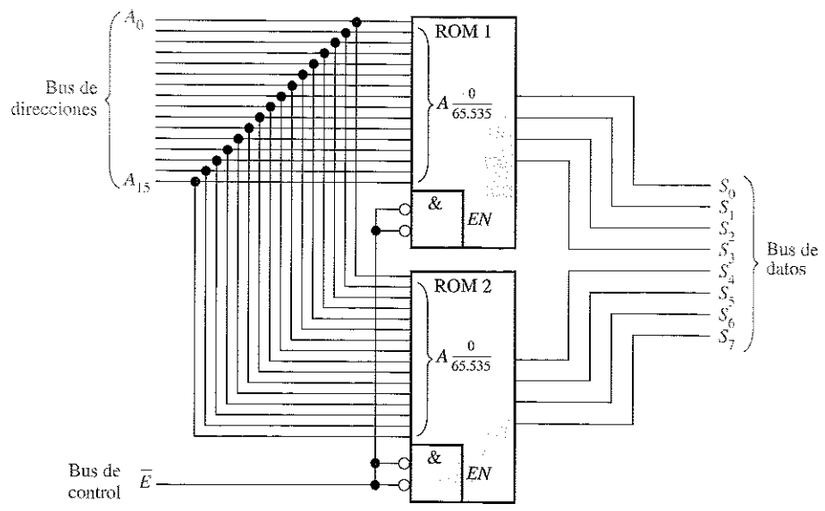


FIGURA 12.40

Problema relacionado. Describir la manera de expandir una ROM de 64k x 1 para obtener una ROM de 64k x 8.

EJEMPLO 12.3 Utilizar las memorias del Ejemplo 12.2 para formar una ROM de 64k x 16.

Solución. En este caso se necesita una memoria que almacene 65.356 palabras de 16 bits. Se requieren cuatro ROM de 64k x 4 para realizar el trabajo, como se muestra en la Figura 12.41.

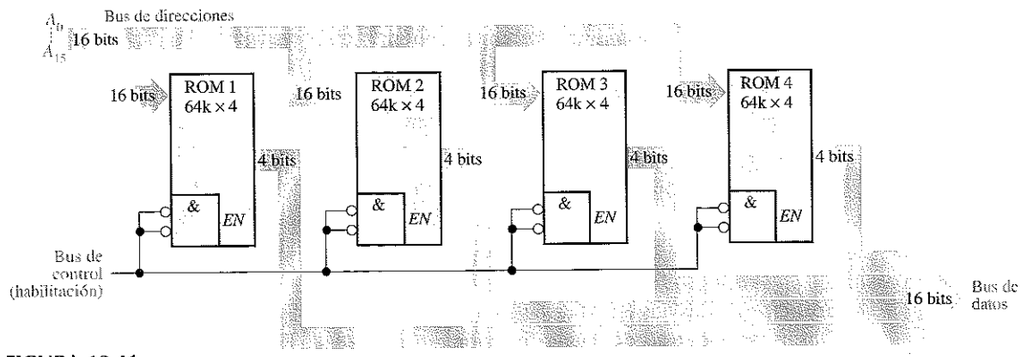


FIGURA 12.41

Problema relacionado. ¿Cuántas ROM de 64k x 1 se requerirían para implementar la memoria mostrada en la Figura 12.41?

Una ROM sólo tiene salidas de datos, pero una RAM tiene entradas y salidas de datos. Para realizar la expansión de la longitud de palabra en una RAM (SRAM o DRAM), las entradas *y* salidas de datos forman el bus de datos. Puesto que las líneas de entrada de datos y las correspondientes líneas de salida de datos deben conectarse juntas, se requieren buffers triestado. La mayoría de las RAM proporciona circuitería de tres estados interna. La Figura 12.42 ilustra la expansión de la RAM para incrementar la longitud de palabra.

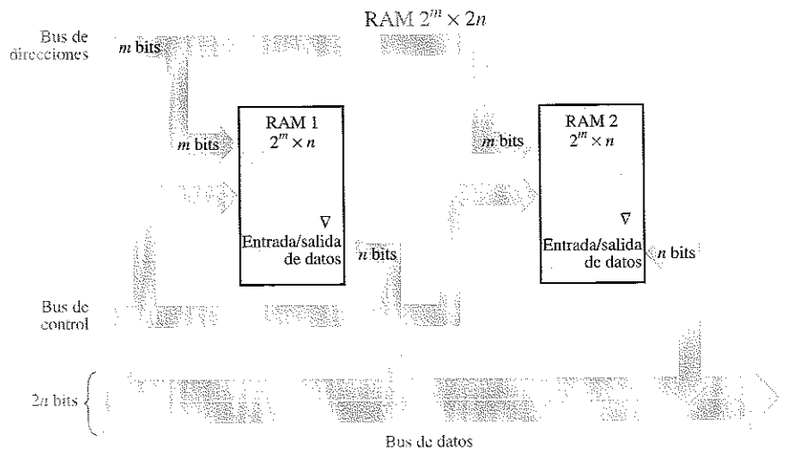


FIGURA 12.42
Ilustración de la expansión de palabra, con dos RAM de $2^m \times n$ unidas para formar una RAM de $2^m \times 2n$.

EJEMPLO 12.4 Utilizar memorias SRAM de 1M x 4 para crear una SRAM de 1M x8.

Solución. Se conectan dos SRAM de 1M × 4 como muestra el diagrama de bloques simplificado de la Figura 12.43.

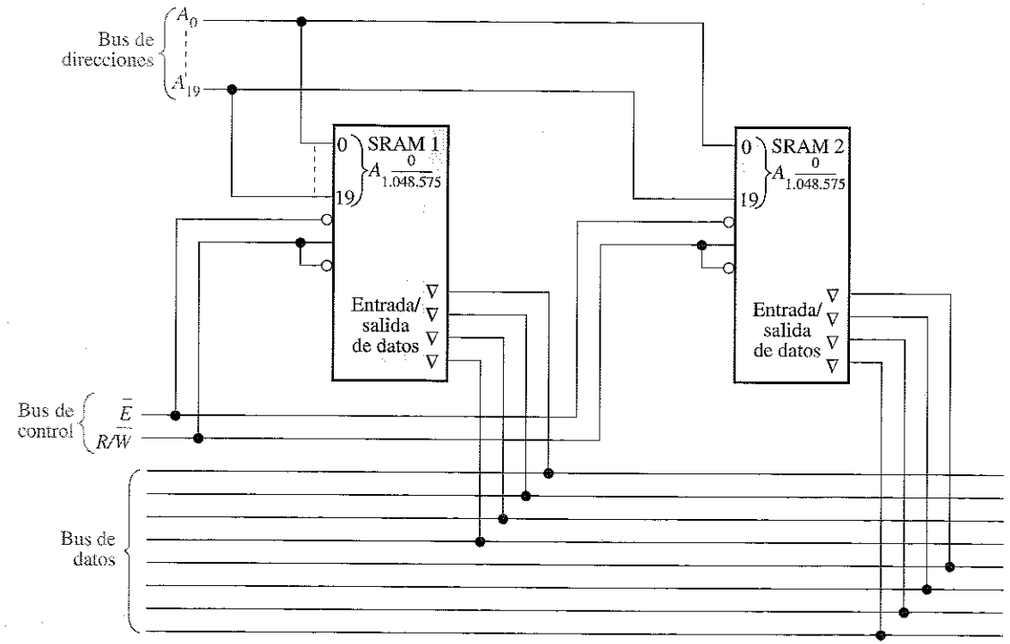


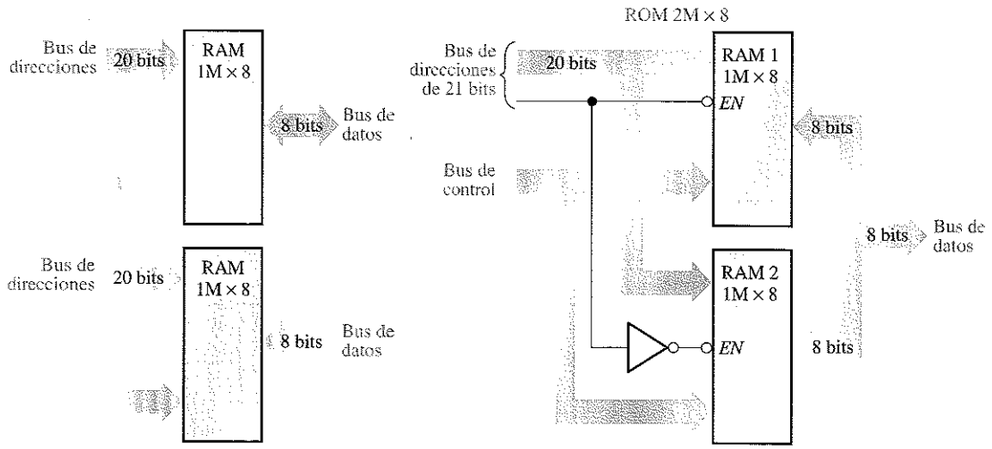
FIGURA 12.43

Problema relacionado. Utilizar memorias SRAM de 1M x 8 para crear una SRAM de 1M x 16.

Expansión de la capacidad de palabra

Cuando las memorias se amplían para incrementar la **capacidad de palabra**, el *número de direcciones se aumenta*. Para conseguir este incremento, el número de bits de dirección se debe aumentar como se ilustra en la Figura 12.44, en la que se expanden dos RAM de 1M x 8 para formar una memoria de 2M x 8.

Cada memoria individual tiene 20 bits de dirección para seleccionar 1.048.576 direcciones, como se muestra en la parte (a). La memoria expandida tiene 2.097.152 direcciones y, por tanto, requiere 21 bits de dirección, como se muestra en la parte (b). El bit de dirección número 21 se utiliza para activar el chip de memoria adecuado. El bus de datos de la memoria expandida resultante sigue siendo de ocho bits. En el Ejemplo 12.5 se ilustran los detalles de esta expansión.



(a) Cada memoria individual almacena 1.048.576 palabras de 8 bits

(b) Las memorias se expanden para formar una RAM de 2M x 8 que requiere un bus de direcciones de 21 bits

FIGURA 12.44
Expansión de la capacidad de palabra.

EJEMPLO 12.5 Utilizar memorias RAM de 512k x 4 para implementar una memoria de 1M x 4.

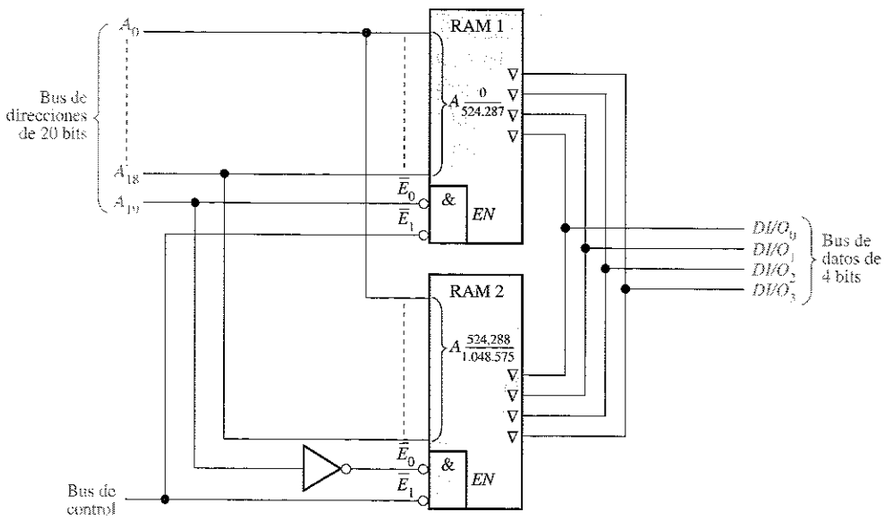


FIGURA 12.45

Solución. La expansión del direccionamiento se consigue conectando la entrada de habilitación de chip ($\overline{E_0}$) al vigésimo bit de dirección (A_{19}) como muestra la Figura 12.45.

La entrada $\overline{E}_1$ se usa como entrada de habilitación común a las dos memorias. Cuando el vigésimo bit de dirección (A_{19}) está a nivel BAJO, se selecciona la RAM 1 (la RAM 2 está desactivada), y los 19 bits de dirección de menor orden (A_0-A_{18}) permiten acceder a las direcciones de la RAM 1. Cuando el vigésimo bit de dirección (A_{19}) está a nivel ALTO, la RAM 2 se activa debido al nivel BAJO en la salida del inversor (la RAM 1 está desactivada), y los 19 bits de dirección de menor orden (A_0-A_{18}) permiten acceder a cada una de las direcciones de la RAM 2.

Problema relacionado. ¿Cuáles son los rangos de direcciones de la RAM 1 y de la RAM 2 de la Figura 12.45?

Consejos Prácticos

Los componentes de memoria son extremadamente sensibles a la electricidad estática. Tome las siguientes precauciones cuando esté manejando chips de memoria o módulos de tipo SIMM o DIMM:

- ❑ Antes de manejar los componentes, descargue la electricidad estática de su cuerpo tocando una superficie puesta a masa, o lleve una muñequera de puesta a tierra con una resistencia de alto valor, si dispone de una. Una toma de tierra cómoda y fiable es la que puede encontrarse en los enchufes eléctricos.
- ❑ No extraiga los componentes de sus bolsas antiestáticas hasta que esté listo para instalarlos.
- ❑ No deposite componentes sobre las bolsas antiestáticas, porque sólo el interior de las mismas es antiestático.
- ❑ Cuando esté manejando módulos SIMM o DIMM, sosténgalos por los bordes o por la abrazadera metálica de montaje. No toque los componentes de las placas ni los terminales del conector de borde.
- ❑ No deslice nunca un componente sobre ningún tipo de superficie.
- ❑ Evite la presencia de plástico, vinilo, espuma de poliestireno y nylón en el área de trabajo.

Módulos SIMM y DIMM

Las memorias se suelen suministrar en forma de módulos de memoria de terminal simple (single in-line memory module, **SIMM**) o como módulos de memoria de terminal doble (dual in-line memory module, **DIMM**). Los módulos SIMM y los más recientes módulos DIMM son pequeñas tarjetas de circuito impreso en las que se montan chips de memoria con las entradas y salidas conectadas a un conector de borde situado en la parte inferior de la tarjeta de circuito. Los módulos DIMM son generalmente más rápidos, pero sólo pueden ser instalados en las máquinas más modernas, que hayan sido diseñadas para admitirlos.

Los módulos SIMM se clasifican en módulos de 30 y de 72 pines. Ambos tipos se ilustran en la Figura 12.46. Aunque las capacidades de memoria disponibles para los módulos SIMM pueden variar entre 256 KB y 32 MB, la diferencia fundamental entre las dos configuraciones de pines es el tamaño del bus de datos. Generalmente,

los módulos SIMM de 30 pines están diseñados para buses de datos de 8 bits, necesitándose más módulos SIMM para poder manejar más bits de datos. Los módulos SIMM de 72 pines admiten un bus de datos de 32 bits, por lo que hacen falta un par de módulos SIMM para los buses de datos de 64 bits.

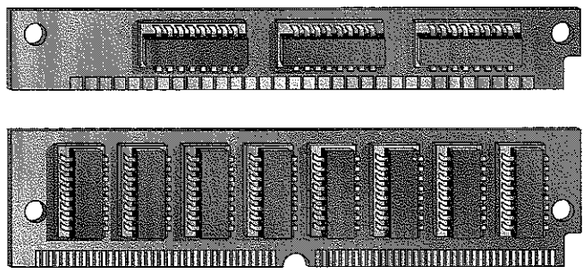


FIGURA 12.46
Módulos SIMM de 30 y de 72 pines.

Los módulos DIMM tienen un aspecto similar a los módulos SIMM, pero proporcionan una mayor densidad de memoria con sólo un incremento relativamente pequeño en el tamaño físico. La diferencia clave estriba en que los módulos DIMM distribuyen los pines de entrada y salida en ambos lados de la tarjeta de circuito impreso, mientras que los módulos SIMM sólo emplean uno de los lados. Las configuraciones normales de los módulos DIMM son las de 72 pines, 100 pines, 144 pines y 168 pines, con las que se pueden emplear buses de datos de 32 y de 64 bits. Por regla general, la capacidad de los módulos DIMM va de 4 a 512 MB.

Los módulos SIMM y DIMM se insertan en zócalos situados en una tarjeta de sistema, como los ilustrados en la Figura 12.47, siendo lo normal que haya varios zócalos disponibles para la expansión de memoria. Por supuesto, los zócalos para los módulos SIMM y DIMM son diferentes y no son intercambiables.

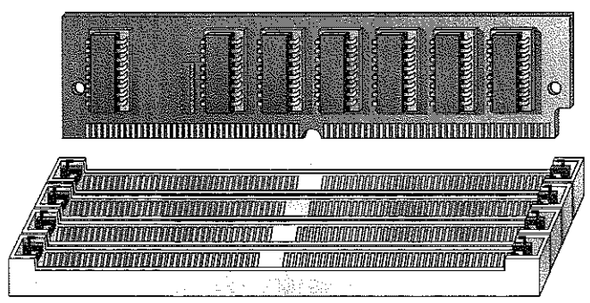


FIGURA 12.47
Un módulo SIMM/DIMM insertado en un zócalo de una tarjeta de un sistema.

Consejos Prácticos

Cuando se instalan módulos SIMM o DIMM, deben observarse siempre las precauciones citadas en los consejos prácticos anteriormente mencionados. A continuación se enumeran los pasos que deben seguirse:

1. Alinee las muescas de la tarjeta del módulo SIMM o DIMM con las muescas del zócalo de memoria.
2. Empuje firmemente sobre el módulo hasta que encaje perfectamente en el zócalo.
3. Generalmente, los cierres en ambos lados del zócalo se ajustarán al insertar completamente el módulo. Estos cierres también permiten liberar el módulo, para poderlo sacar del zócalo.

REVISIÓN DE LA SECCIÓN 12.6

1. ¿Cuántas RAM de 16k × 1 se requieren para conseguir una memoria con una capacidad de palabra de 16k y una longitud de palabra de ocho bits?
2. Para expandir la memoria de 16k × 8 de la cuestión anterior en una organización de 32k × 8, ¿cuántas RAM de 16k × 1 se requieren?
3. ¿Qué significa SIMM?
4. ¿Qué significa DIMM?

12.7 ■ TIPOS ESPECIALES DE MEMORIAS

En esta sección, se cubren la memoria FIFO (first in-first out, primero en entrar primero en salir), la memoria LIFO (Last in-first out, último en entrar primero en salir), la pila de memoria y el dispositivo de memoria de acoplamiento carga.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir una memoria FIFO. ■ Describir una memoria LIFO. ■ Utilizar las pilas de memoria. ■ Explicar cómo se usa una parte de una RAM como pila de memoria. ■ Describir una memoria básica CCD.

Memorias FIFO (First In-First out)

Este tipo de memoria está formado por una disposición de registros de desplazamiento. El término **FIFO** hace referencia al funcionamiento básico de este tipo de memoria, en la que el primer bit de datos que se escribe es el primero que se lee.

En la Figura 12.48 se ilustra una diferencia importante entre un registro de desplazamiento convencional y un registro FIFO. En un registro convencional, un bit de datos se desplaza a través del registro sólo cuando se introducen nuevos datos; en un registro FIFO, un bit de datos atraviesa el registro hasta situarse en la posición de bit más a la derecha que esté vacía.

La Figura 12.49 es el diagrama de bloques de una memoria serie FIFO. Esta memoria en particular tiene cuatro registros de datos serie de 64 bits y un registro de control de 64 bits (registro de marca). Cuando los datos se introducen mediante un

impulso de desplazamiento de entrada, automáticamente, bajo el control del registro de marca, se mueven a la posición vacía más próxima a la salida. Los datos no pueden avanzar a las posiciones que están ocupadas. Sin embargo, cuando un bit de datos se desplaza mediante un impulso de desplazamiento de salida, los bits de datos que están en los registros automáticamente se mueven a la posición siguiente hacia la salida. En una memoria FIFO asíncrona, los datos se desplazan hacia fuera independientemente de la entrada de datos, utilizando dos relojes separados.

Registro de desplazamiento convencional

Entrada	X	X	X	X	Salida
0	0	X	X	X	→
1	1	0	X	X	→
1	1	1	0	X	→
0	0	1	1	0	→

X = bits de datos desconocidos.

En un registro de desplazamiento convencional, los datos permanecen a la izquierda hasta que son desplazados por medio de datos adicionales

Registro de desplazamiento FIFO

Entrada	—	—	—	—	Salida
0	—	—	—	0	→
1	—	—	1	0	→
1	—	1	1	0	→
0	0	1	1	0	→

— = posiciones vacías.

En un registro de desplazamiento FIFO, los datos "van cayendo" (a la derecha)

FIGURA 12.48
Comparación del funcionamiento de un registro convencional y uno FIFO.

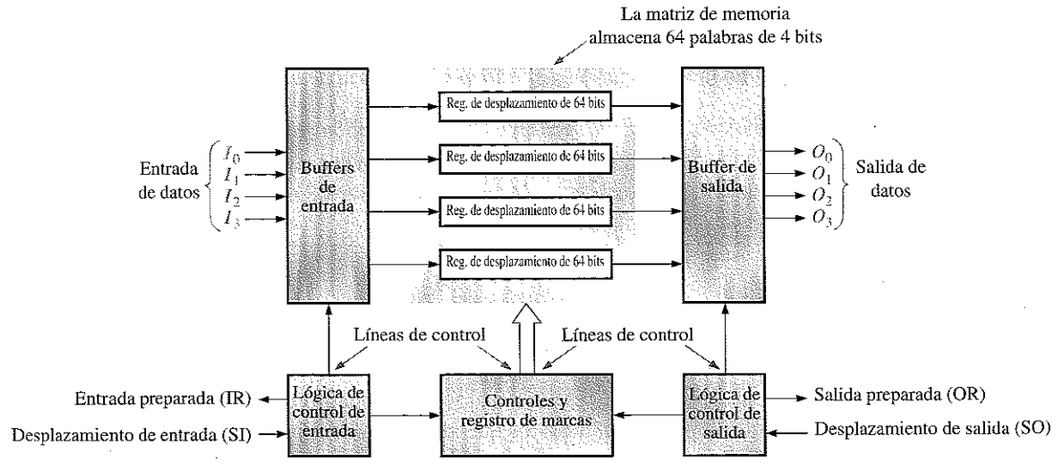
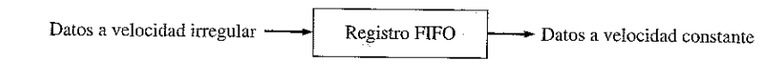


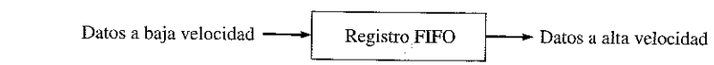
FIGURA 12.49
Diagrama de bloques de una memoria FIFO serie típica.

Aplicaciones de una FIFO

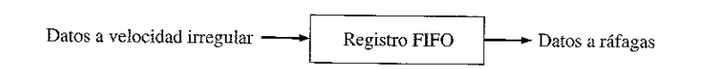
Un área de aplicación importante del registro FIFO es el caso en que dos sistemas con velocidades diferentes tienen que comunicarse. Los datos pueden entrar en un registro FIFO a una velocidad y salir a otra velocidad distinta. La Figura 12.50 muestra cómo debe emplearse un registro FIFO en estas situaciones.



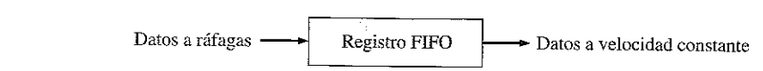
(a) Datos irregulares de telemetría pueden almacenarse y retransmitirse a velocidad constante.



(b) Los datos de entrada a velocidad lenta de un teclado pueden almacenarse y transferirse a una velocidad mayor para su procesamiento.



(c) Los datos de entrada a velocidad continua se pueden almacenar y luego pueden salir a ráfagas.



(d) Los datos a ráfagas se pueden almacenar y reformatear para que salgan a velocidad continua.

FIGURA 12.50
Ejemplos de registro FIFO para aplicaciones de control de la velocidad de transmisión de los datos.

Memorias LIFO (Last in-First out)

Las memorias **LIFO** se encuentran en aplicaciones que utilizan microprocesadores y otros sistemas de computación. Permiten almacenar datos y luego extraerlos en orden inverso; es decir, el último byte de datos almacenado es el primer byte de datos que se recupera.

Pilas de registros. Comúnmente, una memoria LIFO se denomina pila push-down. En algunos sistemas, se implementa con un grupo de registros, como muestra la Figura 12.51. Una pila puede estar formada por cualquier número de registros, pero el registro superior se denomina *tope de la pila.*

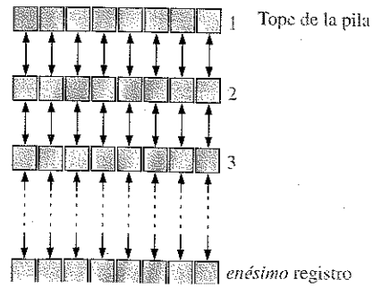


FIGURA 12.51
Pila de registros.

Para ilustrar el principio de estas memorias, se carga en paralelo un byte de datos en el tope de la pila. Cada sucesivo byte empuja al anterior al registro siguiente. Este proceso se muestra en la Figura 12.52. Obsérvese que cada nuevo dato siempre se carga en el registro superior, y los bytes previamente almacenados son empujados hacia abajo dentro de la pila. El nombre de *pila push-down* (pila de empuje hacia abajo) viene de esta característica.

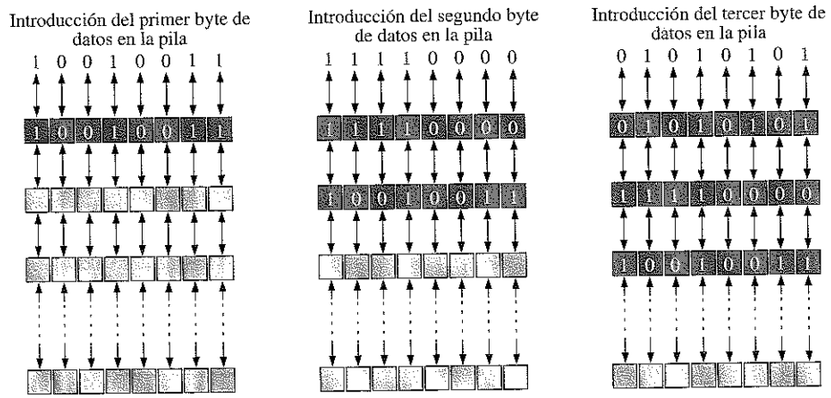


FIGURA 12.52
Ilustración simplificada del proceso de introducción de datos en la pila.

Los bytes de datos se recuperan en orden inverso. El último byte introducido siempre está en el registro superior de la pila, por lo que, cuando sale de la pila, los demás bytes saltan a las siguientes posiciones superiores. Este proceso se ilustra en la Figura 12.53.

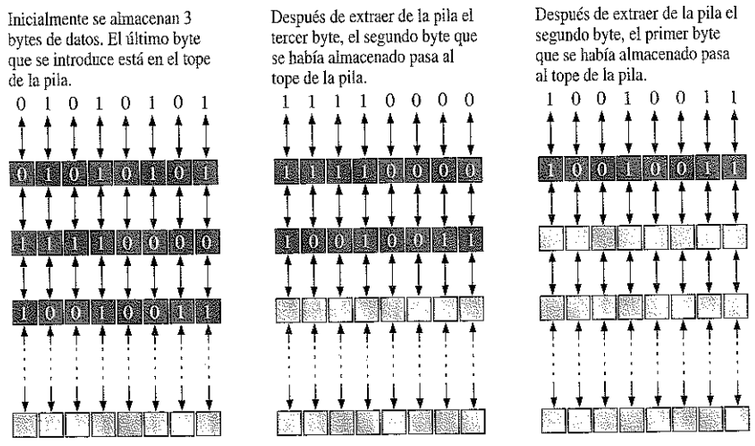
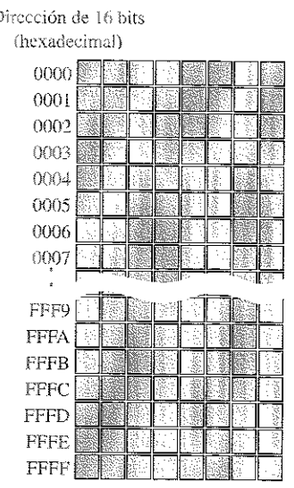


FIGURA 12.53
Ilustración simplificada de extracción de datos de la pila.

Pila RAM. Otra forma de implementar una memoria LIFO en algunos sistemas basados en microprocesador, es disponer una sección de la RAM como pila, mejor que utilizar un conjunto de registros dedicados. Como hemos visto, en un registro de pila, los datos se desplazan de arriba a abajo desde una posición a la siguiente. En una pila RAM, los datos en sí no se desplazan, sino que es el tope de la pila lo que se mueve bajo el control de un registro denominado puntero de pila.

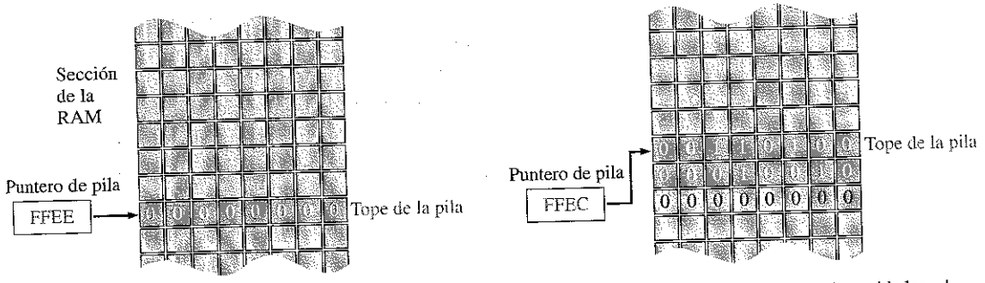
Consideremos una memoria de acceso aleatorio organizada en bytes, es decir, en la que cada dirección contiene ocho bits, como muestra la Figura 12.54. La dirección binaria 0000000000001111, por ejemplo, puede escribirse en hexadecimal como 000F. Una dirección de 16 bits puede tener un valor hexadecimal *mínimo* de 0000_{16} y un valor *máximo* de $FFFF_{16}$. Utilizando esta notación, una matriz de memoria de 64 kbytes se puede representar como se muestra en la Figura 12.54. La dirección de memoria más baja es 0000_{16} y la dirección más alta es $FFFF_{16}$.

FIGURA 12.54
Representación de una memoria de 64 kbytes con direcciones de 16 bits expresadas en hexadecimal.



Ahora, supongamos que se reserva una sección de la RAM para utilizarla como pila. Un registro independiente especial, denominado puntero de pila, contiene la dirección del registro superior de la pila, tal y como se muestra en la Figura 12.55(a). Se utiliza una representación hexadecimal de 4 bits para las direcciones binarias. En la figura, las direcciones se han elegido arbitrariamente, con el único fin de servir de ilustración.

Ahora vamos a ver cómo se introducen los datos en la pila. Inicialmente, el puntero de pila se encuentra en la dirección $FFEE_{16}$, que es el tope de la pila, como se indica en la Figura 12.55(a). El puntero de pila se decrementa en dos unidades y toma el valor $FFEC_{16}$. Esto hace que el tope de la pila se mueva a una dirección de memoria inferior, como se ve en la Figura 12.55(b). Obsérvese que el tope de la pila no es estacionario como en el caso de la pila de registros fijos, sino que se mueve hacia abajo (direcciones más bajas) por la RAM, según se almacenan las palabras de datos. La Figura 12.55(b) muestra dos bytes (una palabra de datos) que se introducen en la pila. Después de que se almacena la palabra, el tope de la pila está en $FFEC_{16}$.

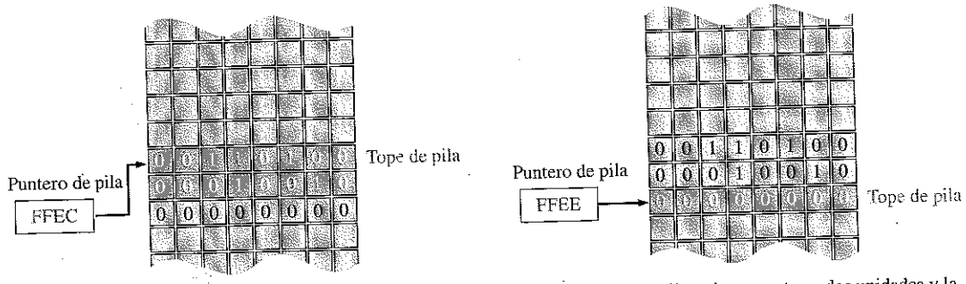


(a) Inicialmente, el puntero de pila está en FFEE antes de que la palabra de datos 0001001000110100 (1234) se introduzca en la pila.

(b) El puntero de pila se decrementa en dos unidades y la palabra de datos 0001001000110100 se coloca en las dos posiciones anteriores al valor original del puntero de pila.

FIGURA 12.55
Ilustración de la operación de introducción de datos en una pila RAM.

La Figura 12.56 ilustra la operación de extracción de los datos de la pila RAM. La última palabra de datos almacenada en la pila se lee en primer lugar. El puntero de pila, que está en la posición FFEC, se incrementa en dos unidades para apuntar a la dirección $FFEE_{16}$ y se realiza una operación de extracción, como puede verse en la parte (b) de la figura. Téngase en mente que las memorias RAM, cuando se leen, no son destructivas, por lo que los datos permanecerán almacenados en la memoria después de la operación de extracción. Una palabra de datos sólo se destruye cuando se escribe sobre ella.



(a) El puntero de pila se encuentra en la posición FFEC antes de que se copie (extraiga) la palabra de datos de la pila.

(b) El puntero de pila se incrementa en dos unidades y la última palabra de datos almacenada se copia (se extrae) de la pila.

FIGURA 12.56
Ilustración de la operación de extracción de datos de la pila RAM.

Una pila RAM puede tener cualquier profundidad, dependiendo de la cantidad de direcciones de memoria asignada para dicho propósito.

Memorias CCD

La memoria **CCD** (charge-coupled device, dispositivo de acoplamiento de carga) almacena los datos como cargas de condensador. Sin embargo, a diferencia de la RAM dinámica, la celda de almacenamiento no incluye un transistor. La principal ventaja de estas memorias CCD es su alta densidad.

La memoria CCD está formada por largas filas de condensadores semiconductores, denominados *canales*. Los datos se introducen en serie en el canal, depositando una pequeña carga en el condensador si se trata de un 0, y una carga grande si es un 1. Después, estas cargas se desplazan a lo largo del canal mientras que se introducen más datos, de acuerdo con las señales de reloj.

Como en el caso de la DRAM, las cargas se deben refrescar periódicamente. Este proceso se realiza desplazando las cargas en serie a través de un circuito de refresco. La Figura 12.57 muestra el concepto básico de un canal CCD. Como los datos se desplazan en serie a través de los canales, estas memorias tienen un tiempo de acceso relativamente grande. Las matrices CCD se usan en algunas cámaras modernas, para captura de imágenes de vídeo en forma de carga de luz inducida.

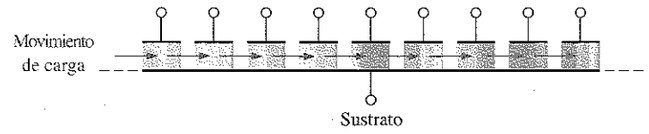


FIGURA 12.57
Un canal CCD (charge-coupled device).

REVISIÓN DE LA SECCIÓN 12.7

1. ¿Qué es una memoria FIFO?
2. ¿Qué es una memoria LIFO?
3. Explicar la operación de introducción de datos en una pila de memoria.
4. Explicar la operación de extracción de datos de una pila de memoria.
5. ¿De qué es abreviatura el término CCD?

12.8 ■ MEMORIAS ÓPTICAS Y MAGNÉTICAS

En esta sección nos ocuparemos de los fundamentos de los discos magnéticos, cintas magnéticas, discos magneto-ópticos y discos ópticos. Estos medios de almacenamiento son muy importantes, especialmente en aplicaciones de computadoras, en las que se utiliza almacenamiento masivo no volátil de datos y programas.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir un disco duro magnético. ■ Describir los discos flexibles. ■ Describir los discos duros extraíbles. ■ Explicar el principio básico de los discos magneto-ópticos. ■ Definir los discos CD-ROM, CD-R y CD-RW. ■ Definir una memoria WORM ■ Describir el DVD-ROM.

Almacenamiento magnético

Disco duro magnético. Las computadoras emplean el disco duro como dispositivo interno de almacenamiento masivo. Los **discos duros** son "placas" rígidas de aleación

de aluminio o de una mezcla de vidrio y cerámica recubiertos con una capa magnética. Hay disponibles unidades de disco duro, con dos tamaños principales de diámetro, 5,25 y 3,5 pulgadas, aunque también existen de 2,5 y 1,75 pulgadas. Las unidades de disco duro se sellan herméticamente para mantener al disco libre de polvo.

Normalmente, se apilan dos o más discos sobre un eje o pivote común, que hace que el conjunto gire a una velocidad de miles de revoluciones por minuto. Existe una separación entre cada disco, con el fin de permitir el montaje de un cabezal de lectura-escritura en el extremo del brazo accionador, como se muestra en la Figura 12.58. Hay un cabezal de lectura-escritura en cada cara del disco, ya que los datos se graban en ambas caras de la superficie del disco. El brazo accionador de la unidad sincroniza todos los cabezales de lectura-escritura para mantenerlos perfectamente alineados cuando se desplazan por la superficie del disco, con una separación de sólo una fracción de milímetro con respecto al disco. Una pequeña partícula de polvo podría hacer que un cabezal se rompiera, dañando como consecuencia la superficie del disco.

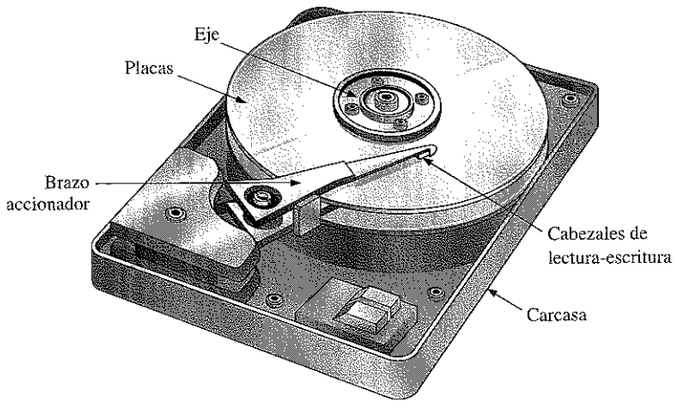


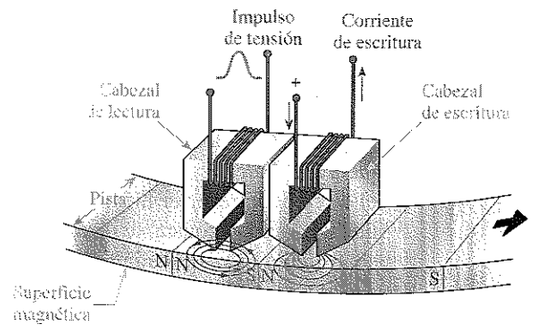
FIGURA 12.58
Esquema simplificado de una unidad de disco duro.

Principios básicos del cabezal de lectura-escritura. El disco duro es un dispositivo de acceso aleatorio, ya que puede recuperar datos almacenados en cualquier lugar del disco, en cualquier orden. En la Figura 12.59 se presenta un diagrama simplificado de la operación de lectura-escritura en la superficie magnética. La dirección o polarización de las partículas magnéticas sobre la superficie del disco se controla mediante la dirección de las líneas de flujo magnético (campo magnético) producidas por el cabezal de escritura, según la dirección de un impulso de corriente en el devanado. Este flujo magnético magnetiza un pequeño punto de la superficie del disco en la dirección del campo magnético. Un punto magnetizado con una cierta polaridad representa un 1 binario, y uno de polaridad opuesta representa un 0 binario. Una vez que se ha magnetizado un punto de la superficie, permanece en dicho estado hasta que se escribe sobre él un campo magnético opuesto.

Cuando un cabezal de lectura pasa por una superficie magnetizada, los puntos magnetizados generan campos magnéticos en el cabezal de lectura, lo que provoca

impulsos de tensión en el devanado. La polaridad de estos impulsos depende de la dirección del punto magnetizado e indica si el bit almacenado es un 1 o un 0. A menudo, los cabezales de lectura y de escritura se combinan en una única unidad.

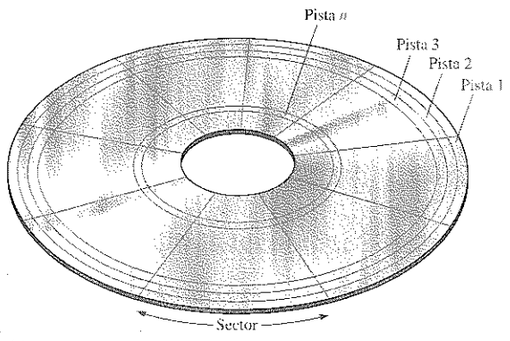
FIGURA 12.59
Operación simplificada de lectura-escritura.



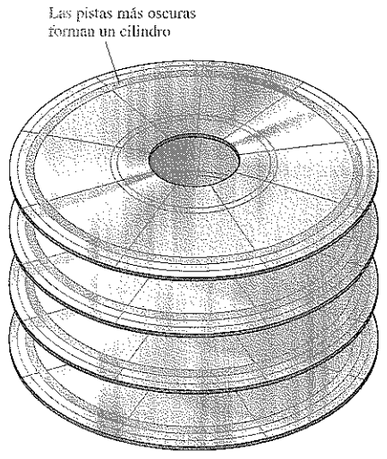
Formato del disco duro. Un disco duro está organizado en pistas y sectores, como muestra la Figura 12.60(a). Cada pista está dividida en una serie de sectores, y cada pista y sector tienen una dirección física que el sistema operativo utiliza para localizar un determinado registro de datos. Normalmente, los discos duros tienen desde unos pocos cientos hasta unos pocos miles de pistas. Como puede verse en la figura, hay un número constante de pistas por sector, utilizando los sectores externos una superficie mayor que los sectores internos. La disposición de pistas y sectores en un disco se denomina *formato*.

La Figura 12.60(b) muestra una pila de disco duro. Las unidades de disco duro difieren en el número de placas apiladas, aunque siempre hay un mínimo de dos. El conjunto de todas las pistas correspondientes de cada placa constituyen lo que se conoce colectivamente como cilindro, como se indica en la figura.

FIGURA 12.60
Organización y formato de un disco duro.



(a)



(b)

NOTAS INFORMÁTICAS
Los datos se almacenan en un disco duro en forma de archivos. Llevar la cuenta de la ubicación de los archivos es el trabajo del controlador de dispositivos que gestiona el disco duro (algunas veces denominado BIOS de la unidad de disco duro). El controlador de dispositivo y el sistema operativo de la computadora pueden acceder a dos tablas que se utilizan para controlar la ubicación y los nombres de los archivos. La primera tabla se denomina FAT (File Allocation Table, tabla de asignación de archivos). La FAT muestra qué partes se han asignado a archivos específicos y mantiene un registro de los sectores abiertos y los sectores defectuosos. La segunda tabla es el directorio raíz, que contiene el nombre de los archivos, el tipo de archivo, la fecha y hora de creación, el número del cluster inicial y otras informaciones referentes al archivo.

Funcionamiento del disco duro. El rendimiento de una unidad de disco duro específica viene determinado por varios parámetros básicos. Una operación de *búsqueda* consiste en el movimiento del cabezal de lectura-escritura hasta la pista deseada. El **tiempo de búsqueda** es el tiempo medio de realización de dicha operación. Normalmente, las unidades de disco duro tienen un tiempo medio de búsqueda, aproximadamente, entre 7 ms y 14 ms, dependiendo de cada unidad concreta.

El **periodo de latencia** es el tiempo que tarda el sector deseado en colocarse debajo del cabezal, una vez que éste se ha posicionado en la pista deseada. El caso peor se produce cuando el sector deseado acaba de pasar la posición del cabezal y está girando, alejándose de la misma. El sector tiene que girar casi una revolución completa hasta alcanzar la posición del cabezal. El *periodo de latencia medio* supone que el disco debe recorrer media revolución. Obviamente, el periodo de latencia depende de la velocidad constante de rotación del disco. Las velocidades de rotación de disco son diferentes para distintas unidades de disco aunque, típicamente, son 3.600 rpm, 4.500 rpm, 5.400 rpm y 7.200 rpm. Algunas unidades de disco recientes giran a una velocidad de 10.033 rpm y tienen un periodo medio de latencia inferior a 3 ms.

La suma del tiempo medio de búsqueda y del periodo medio de latencia es el *tiempo de acceso* de la unidad de disco.

Discos flexibles. El nombre de disco flexible se debe a que este tipo de discos está hecho de un material de poliéster flexible, cubierto por ambas caras con una capa magnética. Los primeros discos flexibles tenían un diámetro de 5,25 pulgadas y estaban contenidos en una funda flexible. Los actuales **discos flexibles** o disquetes tienen un diámetro de 3,5 pulgadas y disponen de una funda de plástico rígido, como se muestra en la Figura 12.61. Una puerta con muelle cubre la ventana de acceso, y permanece cerrada hasta que el disquete se introduce en la unidad. El disco dispone de una placa metálica con un agujero para centrar el disquete y otro para hacerlo rotar dentro de la funda protectora. Los discos flexibles se formatean en pistas y sectores de forma similar a los discos duros, excepto por el número de pistas y sectores. Los disquetes de alta densidad de 1,44 Mbyte tienen 80 pistas por cada lado, con 18 sectores. Con la llegada de otros tipos de discos extraíbles, tales como el Zip, el disquete tiene un serio competidor, aunque su bajo coste puede seguir haciéndole competitivo en aplicaciones que requieran poco espacio de almacenamiento.

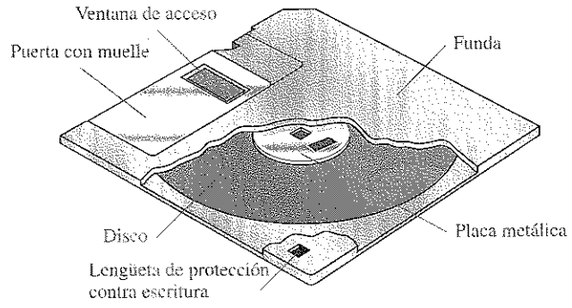


FIGURA 12.61
El disco flexible de 3,5 pulgadas (disquete).

Zip™. La unidad Zip es un tipo de dispositivo de almacenamiento magnético extraíble que parece ser el posible recambio de los disquetes de capacidad limitada. Al igual que el disco flexible, el cartucho del **disco Zip** es un disco flexible incluido en una carcasa rígida aproximadamente del mismo tamaño que el disquete, aunque más grueso. La unidad Zip es mucho más rápida que la unidad de disquetes, dado que tiene una velocidad de giro de 3.000 rpm frente a las 300 rpm de la unidad de disquete. La unidad Zip tiene una capacidad de almacenamiento de 100 Mbytes, lo que es aproximadamente 69 veces mayor que la capacidad de 1,44 Mbytes del disquete. En la Figura 12.62 se muestra una unidad Zip externa normal y su cartucho.

Jaz™. Otro tipo de dispositivo de almacenamiento magnético es la unidad Jaz, que es similar a una unidad de disco duro excepto en que las dos placas se encuentran dentro de un cartucho extraible protegido por un obturador a prueba de polvo. Los **cartuchos Jaz** están disponibles con capacidades de almacenamiento de 1 o 2 Gbytes. La Figura 12.63 muestra un cartucho y una unidad Jaz.

FIGURA 12.62
Unidad y cartuchos Zip™. Cortesía de Iomega Corporation.

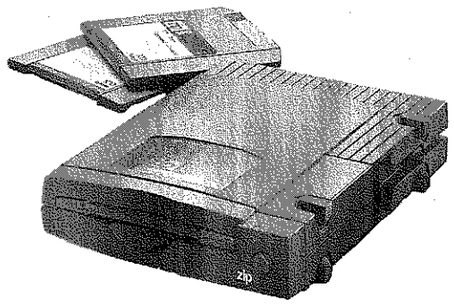


FIGURA 12.63
Unidad y cartuchos Jaz™. Cortesía de Iomega Corporation.

Otras tecnologías de unidades extraíbles

Además de las populares unidades Zip y Jaz, existen otras que merece la pena mencionar. Tenga presente que la tecnología está cambiando de forma tan rápida que probablemente ya habrá más avances en el momento en que se ponga a leer este libro.

La unidad SuperDisk utiliza un disquete de 120 Mbytes que también puede aceptar disquetes de 3,5 pulgadas de 1,44 Mbytes. Asimismo, la unidad SparQ emplea cartuchos de 1 Gbyte y la unidad SyJet admite cartuchos de 1,5 Gbytes. Una diferencia entre la unidad SparQ y la unidad Jaz de 1 Gbyte es que la SparQ es una unidad externa y la interfaz con la computadora se hace a través del puerto paralelo, mientras que la unidad Jaz está disponible tanto en versión externa como interna.

Cinta magnética. La cinta se utiliza para realizar copias de seguridad de datos desde dispositivos de almacenamiento masivo y, normalmente, es mucho más lenta en términos de tiempo de acceso, ya que se accede a los datos en serie en lugar de mediante una selección aleatoria. Hay disponibles diversos tipos de cintas, entre los que se incluyen QIC, DAT, 8 mm y DLT.

QIC es la abreviatura de *quarter-inch cartridge* (cartucho de cuarto de pulgada) y se parece bastante a un casete de audio con dos carretes en el interior. Los distintos estándares QIC varían desde 36 a 72 pistas que pueden almacenar desde 80 Mbytes hasta 1,2 Gbytes. Las actualizaciones más recientes del estándar Travan han aumentado la longitud de la cinta y la anchura de la misma, permitiendo capacidades de almacenamiento de hasta 4 Gbytes. Las unidades de cinta QIC utilizan cabezales de lectura-escritura con un único cabezal de escritura y un cabezal de lectura en cada lado. Esto permite que la unidad de cinta compruebe los datos que se acaban de escribir, cuando la cinta está girando en cualquier dirección. En el modo de registro, la cinta se desplaza bajo los cabezales de lectura-escritura aproximadamente a una velocidad de 100 pulgadas por segundo (25,4 cm/s), como se indica en la Figura 12.64.

DAT, que son las siglas de Digital Audio Tape (cinta digital de audio), utiliza una técnica denominada grabación por exploración helicoidal. Las cintas DAT ofrecen capacidades de almacenamiento en el rango de los 12 Gbytes, pero son más caras que las QIC.

Un tercer tipo de formato de cinta es la cinta de 8 mm que, originalmente, se diseñó para la industria de vídeo, pero que ha sido adoptada por la industria informática como una forma fiable de almacenar grandes cantidades de datos de computadora. La cinta de 8 mm es similar a la DAT, pero ofrece capacidades de almacenamiento de hasta 25 Gbytes.

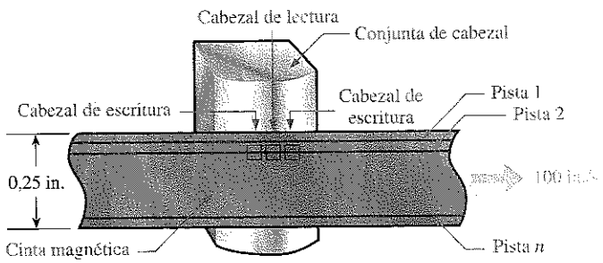


FIGURA 12.64
Cinta QIC.

DLT corresponde a las siglas de Digital Linear Tape (cinta digital lineal). La cinta DLT es una cinta de una anchura de media pulgada (1,25 cm), lo que significa

que es un 60% más ancha que la cinta de 8 mm y, por tanto, dos veces más ancha que una cinta estándar QIC. Básicamente, la cinta DLT difiere en la forma en que trabaja el mecanismo de la unidad de cinta, para minimizar el deterioro de la cinta en comparación con otros sistemas. La cinta DLT ofrece la mayor capacidad de almacenamiento de todos los soportes físicos de este tipo, con capacidades que alcanzan hasta los 35 Gbytes. Un avance reciente en la tecnología DLT promete alcanzar la capacidad de 500 Gbytes.

Almacenamiento magneto-óptico

Como su nombre indica, los dispositivos de almacenamiento magneto-óptico combinan las tecnologías magnética y óptica (láser). Un **disco magneto-óptico** se formatea en pistas y sectores de forma similar a los discos magnéticos.

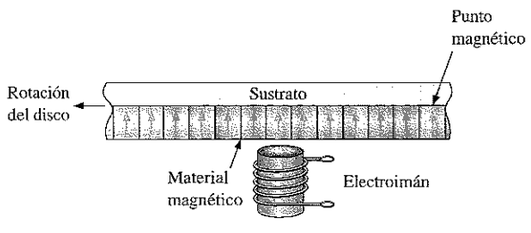
La diferencia básica entre un disco puramente magnético y un disco magneto-óptico es que la capa magnética utilizada en los discos magneto-ópticos requiere calor para alterar la polarización magnética. Por tanto, el disco magneto-óptico es extremadamente estable a temperatura ambiente, haciendo que los datos no cambien. Para escribir un bit de datos, se enfoca un haz láser de alta potencia sobre un punto muy pequeño del disco, y la temperatura de dicho punto se eleva por encima de un nivel de temperatura denominado punto de Curie (aproximadamente 200ºC). Una vez caliente, las partículas magnéticas en dicho punto pueden fácilmente ver cambiada su dirección (polarización) debido al efecto del campo magnético generado por el cabezal de escritura. La información se lee del disco mediante un láser de menor potencia que el que se emplea para escribir, utilizando el efecto de Kerr, según el cual la polaridad de la luz del láser reflejado se altera dependiendo de la orientación de las partículas magnéticas. Los puntos con cierta polaridad representan los ceros y los puntos con la polaridad opuesta representan los unos. El funcionamiento básico magneto-óptico se muestra en la Figura 12.65, que representa una pequeña sección transversal de un disco.

Almacenamiento óptico

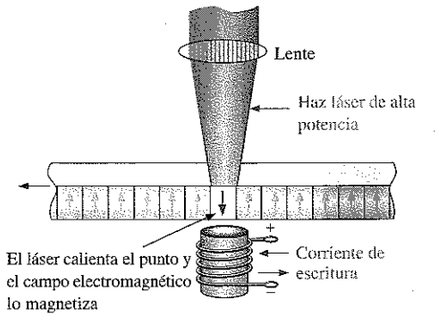
CD-ROM. La memoria de sólo lectura de disco compacto (compact-disc read-only memory) es un disco de 120 mm de diámetro con tres capas dispuestas en forma de sandwich: la capa inferior de plástico de policarbonato, una hoja delgada de aluminio para la reflectividad y una capa superior de laca para protección. El disco **CD-ROM** se formatea con una única pista en forma de espiral, con sectores secuenciales de 2 kbytes y tiene una capacidad de 680 Mbytes. Los datos se pregraban en fábrica en forma de agujeros microscópicos denominados *muescas* y el área plana que rodea a estos agujeros se denomina *planicie*. Las muescas se imprimen en la capa de plástico y no pueden borrarse.

Un reproductor de CD-ROM lee los datos en la pista espiral mediante un haz láser de infrarrojos de baja potencia, como se muestra en la Figura 12.66. Los datos están codificados mediante las muescas y planicies, como muestra la figura. La luz del láser reflejada desde una muesca tiene un desfase de 180º respecto de la luz reflejada desde las planicies. Cuando el disco gira, el estrecho haz de láser incide sobre las muescas y planicies de longitudes variables, y un fotodiodo detecta la diferencia en la

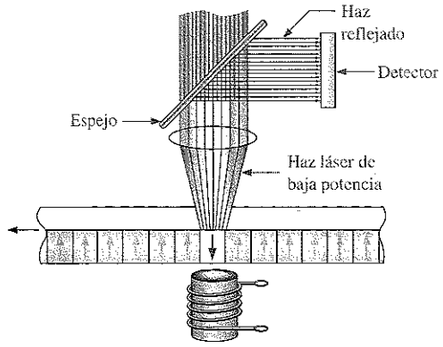
luz reflejada. El resultado es una serie de unos y ceros, que corresponde a la configuración de las muescas y de las planicies a lo largo de la pista.



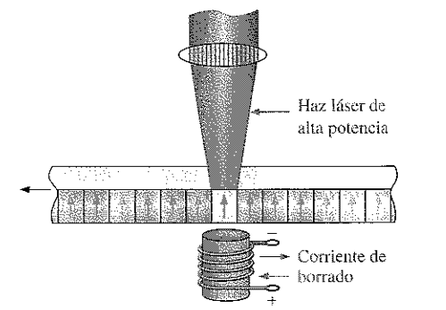
(a) Disco no grabado



(b) Operación de escritura: un haz láser de alta potencia incide sobre un punto, haciendo que las partículas magnéticas se alineen con el campo electromagnético.



(c) Operación de lectura: un haz láser de baja potencia se refleja en las partículas magnéticas de polaridad inversa y su polarización cambia. Si las partículas no están invertidas, la polaridad del haz reflejado no cambia.



(d) Operación de borrado: el campo electromagnético se invierte al mismo tiempo que el haz láser de alta potencia incide en un punto, haciendo que las partículas magnéticas vuelvan a la polaridad original.

FIGURA 12.65
Principio básicos de un disco magneto-óptico.

WORM. Los discos de única escritura-múltiples lecturas (write once-read many) son un tipo de dispositivo de almacenamiento óptico en los que se puede escribir una sola vez, después de lo cual no se pueden borrar los datos, aunque sí se pueden leer muchas veces. Para escribir los datos se utiliza un haz láser de baja potencia, que crea agujeros microscópicos en la superficie del material. Los unos y los ceros quedan representados por las áreas en las que hay y no hay agujero.

CD-R. Prácticamente, es un tipo de WORM. La diferencia se encuentra en que el CD grabable permite múltiples sesiones de escritura en diferentes áreas del disco. El disco **CD-R** tiene una pista en forma de espiral como el CD-ROM, pero, en lugar de hacerse agujeros mecánicamente sobre el disco para representar los datos, el CD-R

emplea un láser para quemar agujeros micróscopicos en una superficie con tinte orgánico. Cuando se calienta más allá de una temperatura crítica con un haz láser durante la operación de lectura, los puntos quemados cambian de color y reflejan menos luz que las áreas no quemadas. Por tanto, los unos y los ceros se representan en un CD-R mediante las áreas quemadas y no quemadas, mientras que un CD-ROM se representan mediante las muescas y las planicies. Al igual que con el CD-ROM, los datos no pueden borrarse una vez que se han escrito.

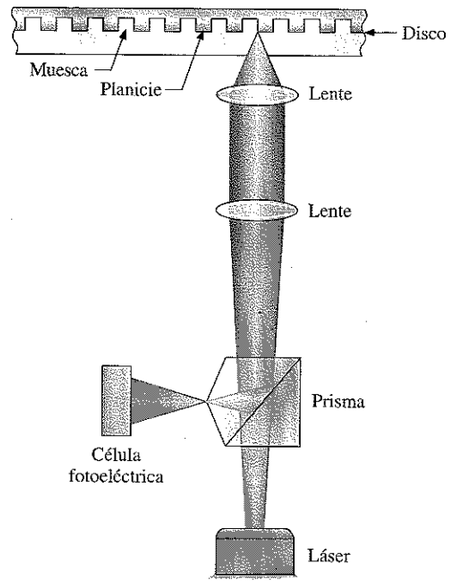


FIGURA 12.66
Operación básica de lectura de datos en un CD-ROM.

CD-RW. El disco CD regrabable puede utilizarse para leer y escribir datos. En lugar de la capa de grabación con tinte que se emplea en el CD-R, el **CD-RW** normalmente utiliza un compuesto cristalino con una propiedad especial. Cuando se calienta a una cierta temperatura, al enfriarse se vuelve cristalino, pero si se calienta a una temperatura superior, se funde y se vuelve amorfo al enfriarse. Para escribir datos, el haz láser enfocado calienta el material a la temperatura de fundido dando lugar al estado amorfo. Las áreas amorfas resultantes reflejan menos la luz que las áreas cristalinas, permitiendo que la operación de lectura detecte los unos y los ceros. Los datos se pueden borrar o sobrescribir calentando las áreas amorfas a una temperatura superior a la temperatura de cristalización, pero inferior a la temperatura de fusión, lo que hace que el material amorfo pase de nuevo al estado cristalino.

DVD-ROM. Originalmente, DVD eran las siglas correspondientes a Digital Video Disk (videodisco digital) pero, actualmente, corresponden a *Digital Versatile Disk*. Al igual que en el CD-ROM, en el **DVD-ROM** los datos se pregraban en el disco. Sin

embargo, el tamaño de las muescas es menor que en el CD-ROM, lo que permite almacenar más datos en una pista. La diferencia principal entre el CD-ROM y el DVD-ROM es que el CD-ROM tiene una única cara mientras que el DVD almacena datos por las dos caras. También, además de los discos DVD de dos caras, hay disponibles discos de múltiples capas que utilizan capas de datos semitransparentes colocadas sobre las capas de datos principales, proporcionando capacidades de almacenamiento de hasta 17 Gbytes. Para acceder a todas las capas, hay que cambiar el enfoque del haz láser para pasar de una capa a otra.

REVISIÓN DE LA SECCIÓN 12.8

1. Enumerar los principales tipos de dispositivos de almacenamiento magnético.
2. ¿Cuál es actualmente la capacidad de almacenamiento de los discos flexibles?
3. Generalmente, ¿cómo está organizado un disco magnético?
4. ¿Cómo se escriben y se leen los datos en un disco magneto-óptico?
5. ¿Cuáles son las capacidades de almacenamiento de los discos Zip y Jaz?
6. Enumerar los tipos de almacenamiento óptico.

12.9 ■ PRUEBAS Y LOCALIZACIÓN DE AVERÍAS

Ya que las memorias pueden contener una gran cantidad de celdas de almacenamiento, comprobar cada una de ellas puede ser un proceso muy largo y frustrante. Afortunadamente, las memorias usualmente se prueban mediante un procedimiento automático realizado con un equipo de pruebas programable, o con la ayuda de un software para comprobación interna del sistema. La mayoría de los sistemas basados en microprocesador proporcionan la comprobación automática de memoria como parte de su software de sistema.

Al finalizar esta sección, el lector deberá ser capaz de:

- Explicar el método de suma de comprobación utilizado para verificar memorias ROM.
- Explicar el método del patrón ajedrezado utilizado para probar memorias RAM.

Comprobación de una ROM

Puesto que las ROM contienen datos conocidos, se puede comprobar la corrección de los datos almacenados leyendo cada palabra de datos de la memoria, y comparándola con la palabra de datos que se sabe que es correcta. En la Figura 12.67 se ilustra una forma de hacer esto. Este procedimiento requiere una ROM de referencia que contenga los mismos datos que la ROM que se va a comprobar. Un equipo de pruebas especial se programa para leer cada dirección de ambas memorias ROM simultáneamente y comparar los contenidos. El organigrama de la Figura 12.68 presenta la secuencia básica.

Método de la suma de comprobación. Aunque el método anterior comprueba cada dirección de la ROM para asegurar la corrección de los datos, tiene la desventaja de que se requiere una ROM de referencia para cada ROM diferente que se desee probar. También puede ocurrir que la ROM de referencia falle, dando lugar a una indicación falsa de error.

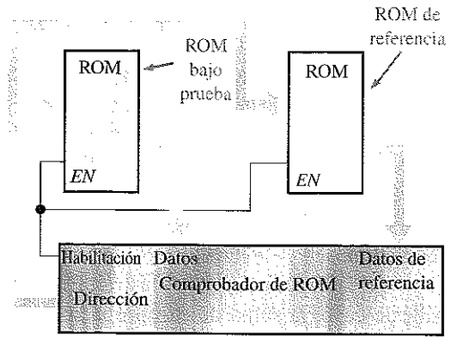
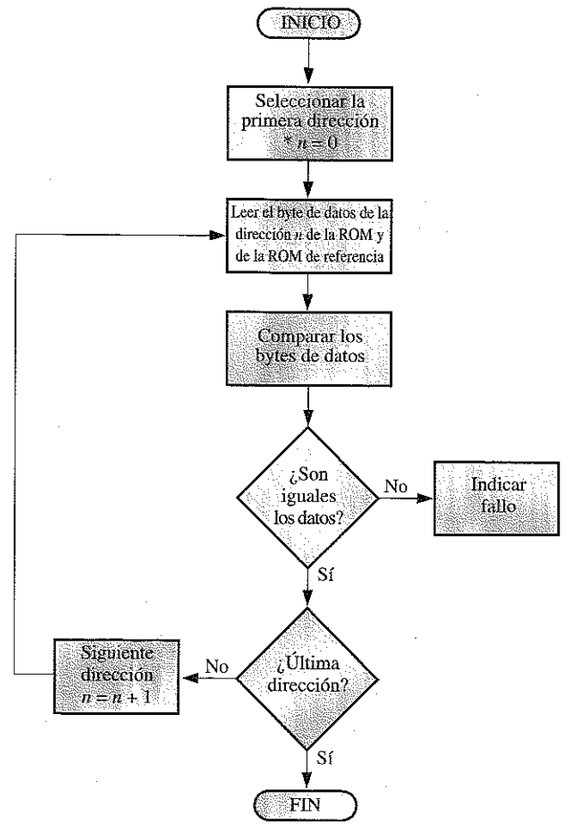


FIGURA 12.67
Diagrama de bloques para la comprobación de todo el contenido de una ROM.



* n es el número de dirección.

FIGURA 12.68
Organigrama para la comprobación de todo el contenido de una ROM.

En el método de suma de comprobación, un número, que es la suma de todos los contenidos de todas las direcciones de la ROM, se almacena en una dirección específica de la memoria cuando ésta se programa. Para probar la ROM, se suman los contenidos de todas las direcciones, excepto la propia suma de verificación, y el resultado se compara con la suma de verificación almacenada en la ROM. Si existe diferencia, es seguro que hay un fallo. Si las sumas de verificación son iguales, muy probablemente la ROM está bien. Sin embargo, existe la remota posibilidad de que una combinación de celdas de memoria erróneas haga que las sumas de verificación sean iguales.

En la Figura 12.69 se ilustra este procedimiento mediante un sencillo ejemplo. En este caso, la suma de comprobación se genera sumando cada columna de bits de datos y descartando los acarreos. Esto equivale a realizar la operación XOR en cada columna. El organigrama de la Figura 12.70 presenta el método de prueba básico mediante la suma de comprobación.

Este método de prueba puede implementarse con un equipo de pruebas especial, o puede incorporarse en una rutina de prueba dentro del software del sistema, en los sistemas basados en microprocesador. En este caso, la rutina de prueba de la ROM se ejecuta automáticamente al arrancar el sistema.

FIGURA 12.69
Ilustración simplificada de una ROM programada, con una suma de comprobación almacenada en una dirección específica.

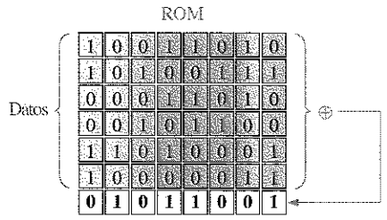
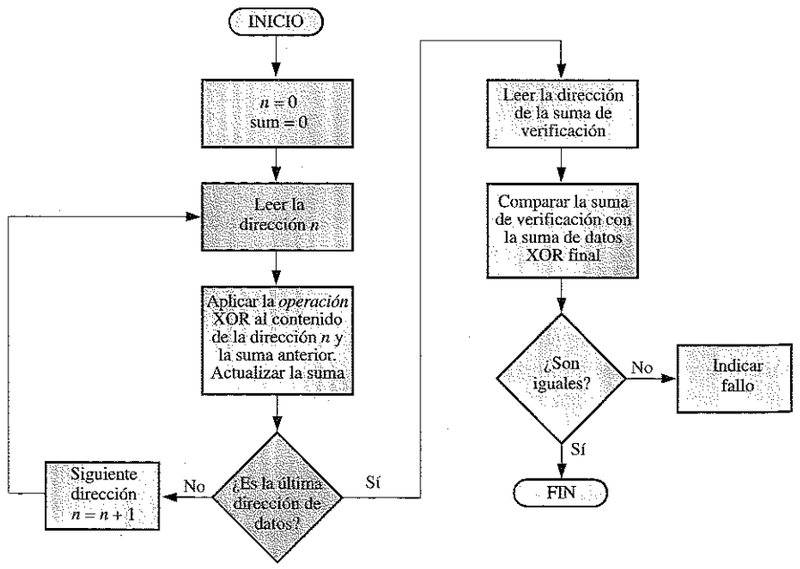


FIGURA 12.70
Organigrama para el método básico de prueba mediante la suma de comprobación.



Comprobación de la RAM

Para probar la capacidad de las memorias RAM de almacenar 0s y 1s en sus celdas se escriben, en primer lugar 0s en todas las celdas de cada dirección, y luego se extraen y verifican. A continuación, se escriben 1s en todas las celdas de cada dirección, y después se leen y verifican. Este método de pruebas básico detectará una celda que se mantenga en el estado 1 o en el estado 0.

Algunos fallos de memoria no se pueden detectar escribiendo ceros en todas las direcciones y unos en todas las direcciones. Por ejemplo, si dos celdas de memoria adyacentes se cortocircuitan, siempre estarán en el mismo estado, siendo ambas 0, o ambas 1. También este método de prueba es inefectivo si existen problemas de ruido interno, consistentes en que los contenidos de una o más direcciones se alteran debido al cambio de los contenidos de otras direcciones.

Método de prueba con patrón ajedrezado. Una forma más completa para probar una RAM consiste en utilizar un patrón de 1s y 0s alternativos, como ilustra la Figura 12.71. Obsérvese que todas las celdas adyacentes contienen bits opuestos. Este patrón comprueba que no haya un cortocircuito entre dos celdas adyacentes, ya que, si existe un cortocircuito, ambas células estarán en el mismo estado.

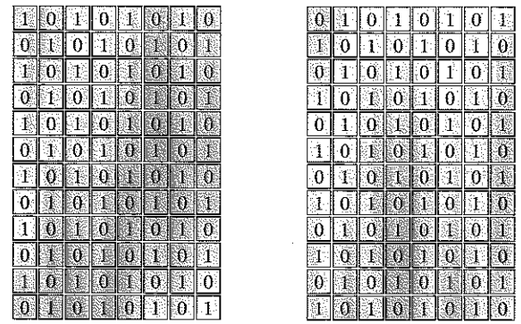


FIGURA 12.71
Patrón de prueba ajedrezado de una RAM.

Después de comprobar la RAM con el patrón de la Figura 12.71(a), éste se invierte, como se muestra en la parte (b) de la figura. Esta inversión verifica la capacidad de todas las celdas de almacenar tanto un 1 como un 0.

Una comprobación más consiste en alternar el patrón en una dirección de memoria cada vez y comprobar que el patrón original no se haya modificado en las restantes direcciones. Esta prueba detectará los problemas consistentes en que los contenidos de una dirección se alteren dinámicamente cuando los contenidos de otra dirección cambien.

En el organigrama de la Figura 12.72 se presenta un procedimiento básico para la prueba del patrón ajedrezado. El procedimiento se puede implementar con el software del sistema en los sistemas basados en microprocesador, de modo que las com-

probaciones se realizan automáticamente cuando se enciende el sistema o se pueden iniciar desde el teclado.

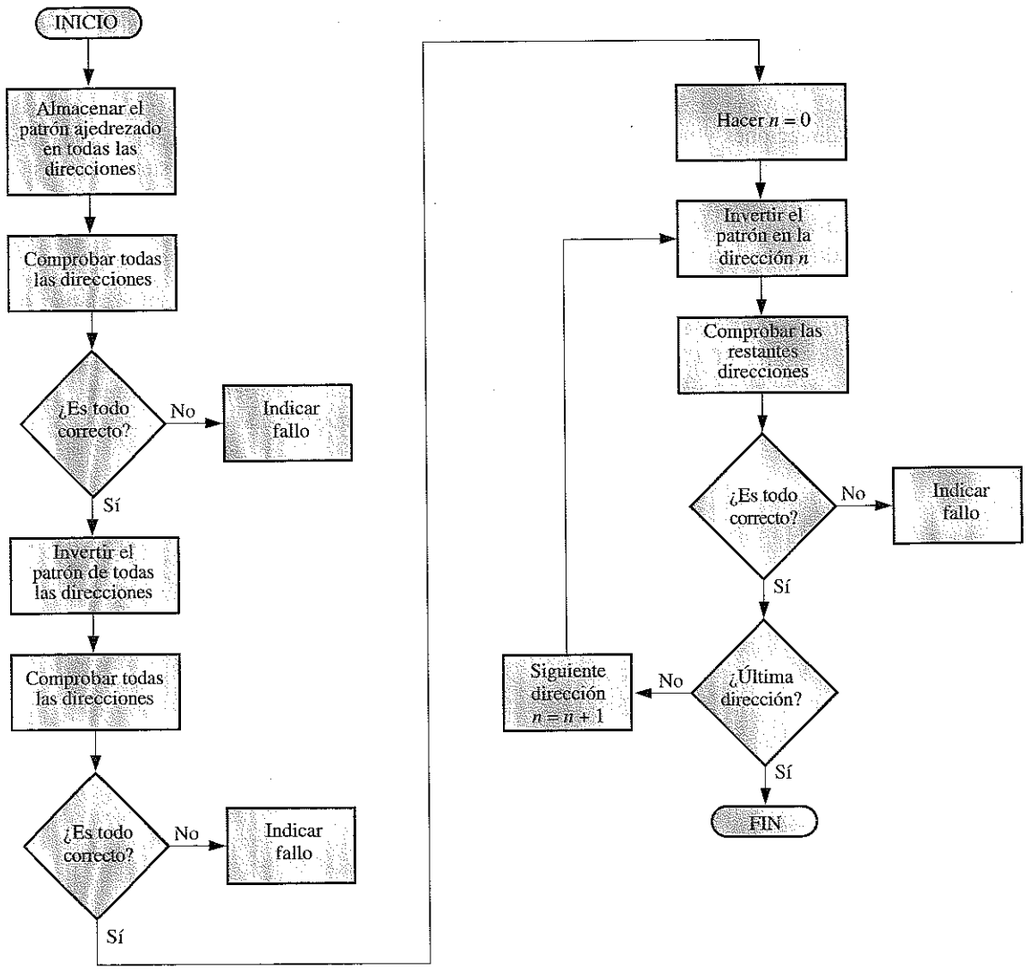


FIGURA 12.72
Organigrama para el método básico de prueba del patrón ajedrezado.

REVISIÓN DE LA SECCIÓN 12.9

1. Describir el método de suma de comprobación para probar una ROM.
2. ¿Por qué no se puede aplicar el método de suma de comprobación para probar una RAM?
3. Enumerar los tres fallos básicos que se pueden detectar en una RAM con la prueba del patrón ajedrezado.

12.10 ■ APLICACIÓN A LOS SISTEMAS DIGITALES

En esta aplicación a los sistemas digitales, se completará el sistema de seguridad de acceso. En el Capítulo 10 se ha desarrollado la lógica para el código de seguridad. En esta sección, se va a desarrollar la lógica de la memoria y se implementará la interfaz entre las dos partes del sistema.

Al finalizar esta sección, el lector deberá ser capaz de:

- Implementar la parte de la memoria del sistema y desarrollar un diagrama lógico.
- Definir la interfaz de la lógica de memoria con la lógica del código de seguridad y con los interruptores externos.
- Explicar el funcionamiento global del sistema.
- Desarrollar un procedimiento de prueba para la tarjeta de memoria.
- Desarrollar un procedimiento de prueba para el sistema completo.

Funcionamiento general del sistema

En la Figura 12.73 se presenta un diagrama de bloques básico del sistema completo de seguridad de acceso. En el Capítulo 10 se ha desarrollado la lógica del código de seguridad y aconsejamos que se revise antes de continuar.

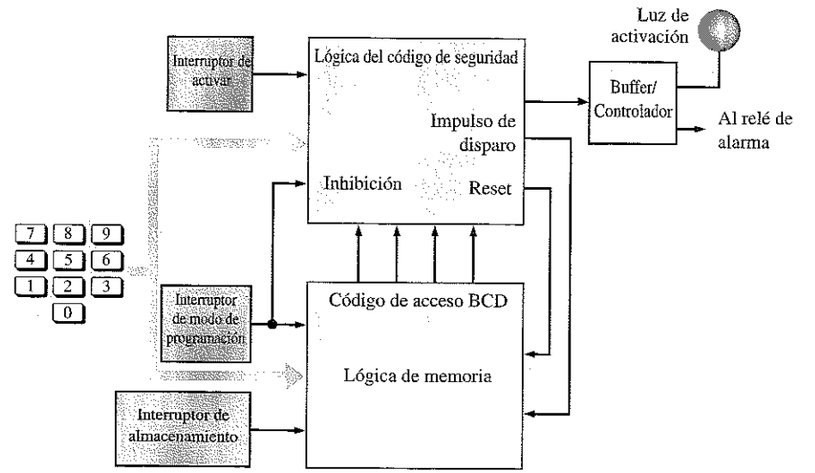


FIGURA 12.73
Diagrama de bloques básico del sistema de seguridad de acceso.

La lógica de memoria utiliza una SRAM para almacenar el código de acceso de 4 dígitos en formato BCD. En el modo de programación, seleccionado mediante un interruptor, se introducen cuatro dígitos en la memoria a través del teclado. Una vez almacenados en la memoria, los cuatro dígitos BCD pasan a ser el código de acceso permanente. Si fuera necesario cambiar el código de acceso, la memoria se reprogramaría con uno nuevo. Mientras que se está en modo de programación, la lógica del código se seguridad se inhibe.

Para programar la memoria, primero se pone el sistema en modo de programación por medio del interruptor de modo de programación, y después se utilizan el interruptor de almacenamiento y el teclado numérico para introducir en memoria el código de cuatro dígitos deseado.

Una vez que se ha programado la memoria con el código deseado, el interruptor de modo de programación se desactiva. Entonces el sistema se habilita presionando el interruptor de activación a la vez que se selecciona cualquier dígito del teclado numérico.

Funcionamiento básico de la lógica de memoria

En la Figura 12.74 se presenta un esquema simplificado de la lógica de memoria. Inicialmente, para almacenar un código en la RAM, se cierra el interruptor de modo de programación, lo que dispara el monoestable B. El interruptor permanece cerrado durante la programación. El impulso de salida del monoestable se aplica a la puerta OR G_2 para poner a cero el contador. En el estado de RESET, el contador contiene todo ceros (000), lo que hace que se seleccione la dirección 0 de la SRAM. La dirección 0 no se utiliza para almacenar códigos.

Para introducir el primer dígito del código de seguridad, se presiona la tecla deseada y se mantiene durante unos pocos segundos mientras que el interruptor de almacenamiento se cierra momentáneamente, disparando el monoestable A. El impulso de salida del monoestable se aplica a la puerta OR G_1, haciendo que el contador pase al estado 001, seleccionando la dirección 1 de la SRAM. Al finalizar el impulso del monoestable A, se dispara el monoestable C y se genera el impulso de escritura en memoria, lo que hace que se almacene el dígito seleccionado en la dirección 1.

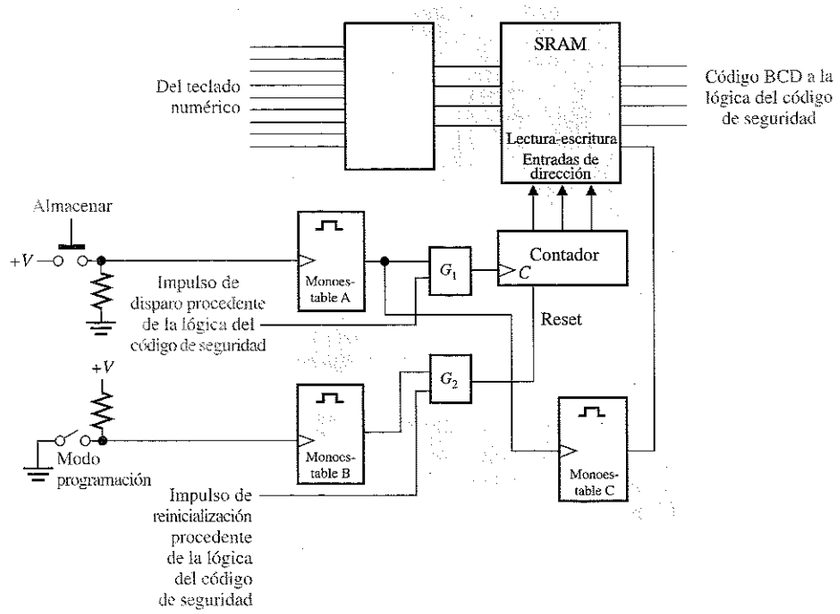


FIGURA 12.74
Diagrama lógico simplificado de la lógica de memoria.

Para introducir el segundo dígito, se mantiene presionada la tecla deseada. De nuevo el monoestable A se dispara y el contador se incrementa pasando al estado 010, seleccionando la dirección 2. Entonces el monoestable C se dispara y genera un impulso de escritura que almacena el dígito del código en la dirección de memoria 2.

Para almacenar el tercer dígito en la dirección 3 de la memoria SRAM y el cuarto dígito en la dirección 4, se repiten los mismos pasos. Una vez que se han almacenado los cuatro dígitos, se abre el interruptor de modo de programación. La SRAM almacena el código mientras que esté alimentada. Se supone que el sistema dispone de una batería de reserva por si falla la alimentación.

■ PRÁCTICAS DE SISTEMAS

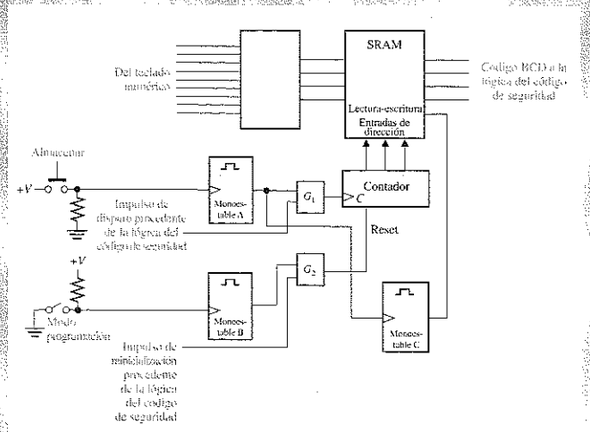
El diagrama lógico básico de la Figura 12.74 indica la lógica general que requiere el sistema, pero no representa en detalle la lógica necesaria para implementarlo. En las siguientes actividades prácticas, analizará el funcionamiento básico de la lógica de memoria para asegurarse de que ha entendido los requisitos funcionales; completará el diseño de la lógica seleccionando los dispositivos adecuados, y luego desarrollará un diagrama lógico detallado que incorpore los dispositivos seleccionados. A continuación, comparará su diagrama lógico con una tarjeta del circuito implementado, y luego desarrollará un procedimiento de pruebas para la tarjeta de la lógica de memoria. Por último, realizará la interfaz entre la tarjeta de la lógica del código de seguridad, la tarjeta de la lógica de memoria y los componentes del panel frontal (interruptores, teclado numérico y luz de activación, con su tarjeta buffer/controladora), para formar el sistema completo.

En la tarjeta de circuito impreso, los taladros de la cara de componentes tienen interconexiones por la parte posterior de la tarjeta. Cuando haga el trazado del circuito en la práctica de sistemas 2, los taladros que están interconectados por la parte posterior se orientan generalmente de forma vertical u horizontal unos respecto a otros, pero no formando ángulo.

■ PRÁCTICA DE SISTEMAS 1: Diseño

Esta práctica está centrada en el diseño de la lógica de memoria. Para obtener información sobre los dispositivos, es necesario recurrir a los capítulos anteriores y/o a las hojas de características.

- ***Actividad 1. Codificador decimal-BCD***
  El teclado numérico utilizado en este sistema produce salidas activas a nivel BAJO. Cuando se presiona una tecla, la línea correspondiente se pone a nivel BAJO y las demás líneas están a nivel ALTO. Especificar un circuito integrado decodificador para establecer la interfaz adecuada con el teclado.
- ***Actividad 2. La SRAM***
  El requisito de memoria de almacenar códigos BCD de cuatro bits establece la longitud de palabra. El requisito de memoria de almacenar cuatro códigos BCD en las direcciones 1 a 4 y el requisito de que la dirección 0 esté reservada para las condiciones de activación o puesta a cero (reset) establece la capacidad de palabra mínima. Una SRAM de capacidad mínima que cumple los requisitos de almacenamiento en términos de su organización es el circuito integrado 74F189. En la Figura 12.75 se proporciona una parte de su hoja de características.
- ***Actividad 3. Contador***
  El contador debe ser capaz de producir una secuencia binaria de 3 bits para direccionar la memoria, y debe disponer de entrada de reset o borrado (clear). Seleccionar un circuito integrado contador adecuado.
- ***Actividad 4. Monoestables***
  El monoestable A sirve para producir un impulso de salida con una anchura de, aproximadamente, 10 ms. Los monoestables B y C sirven para generar cada uno de ellos un impulso de salida con una anchura de, aproximadamente, 1 ms. Seleccionar un circuito integrado monoestable adecuado y determinar los valores de la resistencia y del condensador de cada uno.
- ***Actividad 5. Lógica de puertas***
  En función de la selección que haya hecho para el CI contador y las entradas de la lógica del código de seguridad, especificar los tipos de puertas lógicas (G_1 y G_2) requeridas para proporcionar los niveles lógicos apropiados.
- ***Actividad 6.*** Desarrollar un diagrama lógico detallado utilizando los dispositivos especificados en las actividades anteriores. Mostrar todos los detalles, incluida la numeración de pines.
- ***Actividad 7.*** Escribir un informe técnico detallado que describa el funcionamiento de la lógica de memoria y que especifique la lista de componentes.



54F/74F189 64-Bit Random Access Memory with Tri-State Outputs

General Description

The 'F189 is a high-speed 64-bit RAM organized as a 16-word by 4-bit array. Address inputs are buffered to minimize loading and are fully decoded on-chip. The outputs are TRI-STATE and are in the high impedance state whenever the Chip Select ($\overline{CS}$) input is HIGH. The outputs are active only in the Read mode and the output data is the complement of the stored data.

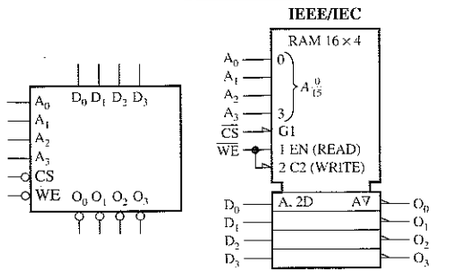
Features

- TRI-STATE outputs for data bus applications
- Buffered inputs minimize loading
- Address decoding on-chip
- Diode clamped inputs minimize ringing
- Available in SOIC, (300 mil only)

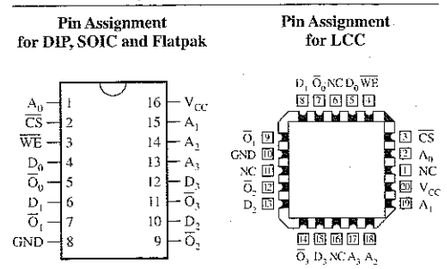
Commercial	Military	Package Number	Package Description
74F189PC		N16E	16-Lead (0.300" Wide) Molded Dual-In-Line
	54F189DL (Note 2)	J16A	16-Lead Ceramic Dual-In-Line
74F189SC (Note 1)		M16A	16-Lead (0.150" Wide) Molded Small Outline, JEDEC
74F189SJ (Note 1)		M16D	16-Lead (0.300" Wide) Molded Small Outline, EIAJ
	54F189FL (Note 2)	W16A	16-Lead Cerpack
	54F189LL (Note 2)	E20A	20-Lead Ceramic Leadless Chip Carrier, Type C

Note 1: Devices also available in 13" reel. Use suffix = SCX and SJX.
Note 2: Military grade device with environmental and burn-in processing. Use suffix = DLQB, FLQB and LLQB.

Logic Symbols



Connection Diagrams



Unit Loading/Fan Out

Pin Names	Description	54F/74F U.L. HIGH/LOW	54F/74F Input I_{IH}/I_{IL}, Output I_{OH}/I_{OL}
A_0–A_3	Address inputs	1.0/1.0	20 μA/ – 0.6 mA
$\overline{CS}$	Chip select input (active LOW)	1.0/1.0	20 μA/ – 1.2 mA
$\overline{WE}$	Write enable input (active LOW)	1.0/1.0	20 μA/ – 0.6 mA
D_0–D_3	Data inputs	1.0/1.0	20 μA/ – 0.6 mA
$\overline{O}_0$–$\overline{O}_3$	Inverted data outputs	150/40 (33.3)	–3.0 mA/24 mA (20 mA)

Function Table

Inputs $\overline{CS}$	Inputs $\overline{WE}$	Operation	Condition of Outputs
L	L	Write	High impedance
L	H	Read	Complement of stored data
H	X	Inhibit	High impedance

H = HIGH voltage level
L = LOW voltage level
X = Immaterial

Block Diagram

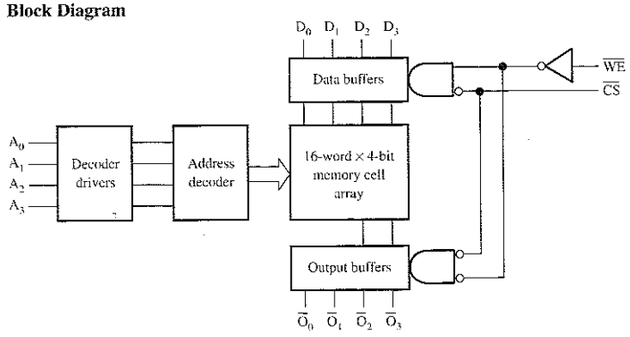
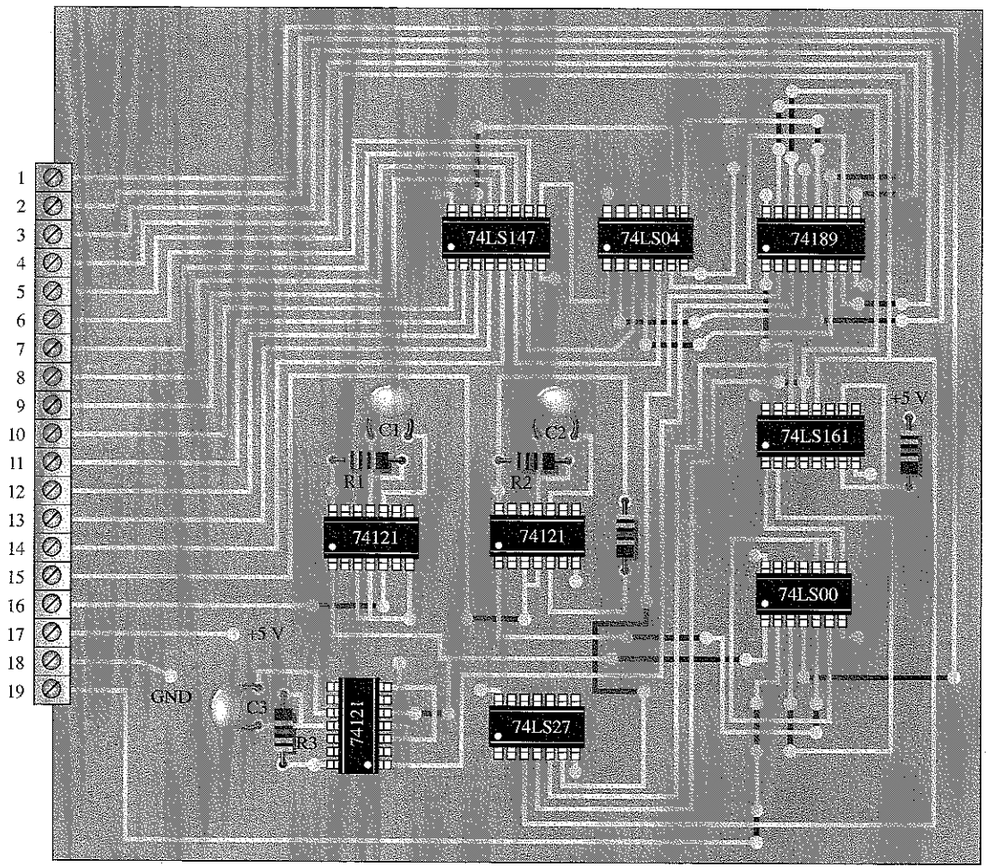


FIGURA 12.75
Hoja de características (parcial) de un SRAM 16 × 4 74F189.

■ PRÁCTICA DE SISTEMAS 2: Verificación y pruebas

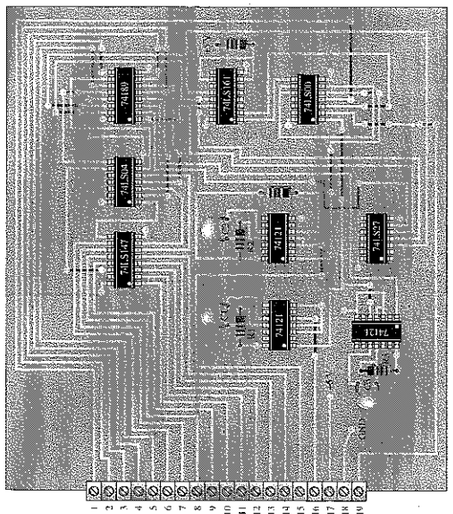
- ***Actividad 1.*** Dibujar el diagrama lógico de la tarjeta de la lógica de memoria abajo mostrada, y compararlo con su diagrama de la actividad 6 de la Práctica de sistemas 1. Comentar las diferencias entre su diseño y el de esta tarjeta y verificar que ambos funcionan.
- ***Actividad 2.*** Identificar las salidas y entradas del conector de la tarjeta.
- ***Actividad 3.*** Desarrollar un procedimiento de pruebas para comprobar la tarjeta mostrada.

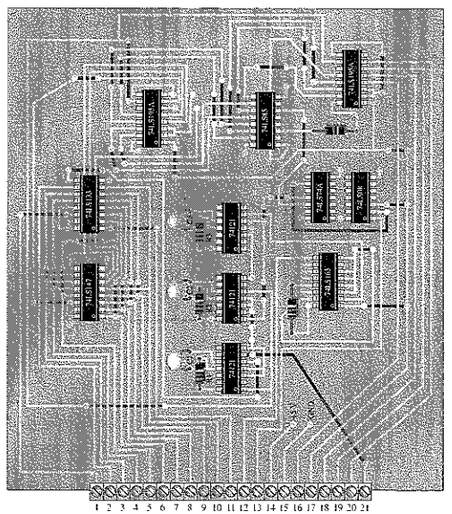


■ PRÁCTICA DE SISTEMAS 3: El sistema completo

Se muestran los componentes y las tarjetas lógicas del sistema de seguridad de acceso. Cuando se interconectan, estos elementos forman el sistema completo de activación y desactivación del sistema de alarma.

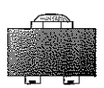
- ***Actividad 1.*** Crear una lista punto a punto para interconectar correctamente todos los componentes y tarjetas.
- ***Actividad 2.*** En el Capítulo 10 y en la Práctica de sistemas 2 de este capítulo, se han desarrollado procedimientos de prueba para comprobar cada tarjeta individualmente. Desarrollar ahora un procedimiento de prueba para el sistema completo, que incluya todas las partes abajo indicadas.
- ***Actividad 3.*** Escribir un informe técnico detallado que describa el funcionamiento global del sistema. Explicar en detalle paso a paso las instrucciones para que el sistema funcione.
- ***Actividad 4.*** Enumerar las posibles razones para los siguientes funcionamientos no válidos del sistema:
  - **(a)** La luz de activación no se enciende cuando se presiona el interruptor ACTIVAR.
  - **(b)** La luz de activación no se enciende cuando se introduce el código correcto.
  - **(c)** La luz de activación se apaga después de introducir el primer dígito del código.





Interruptor de palanca

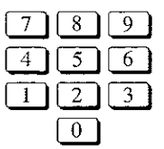
Pulsador A



Pulsador B

Tarjeta buffer/ controladora para el LED de activación y el relé de alarma

LED



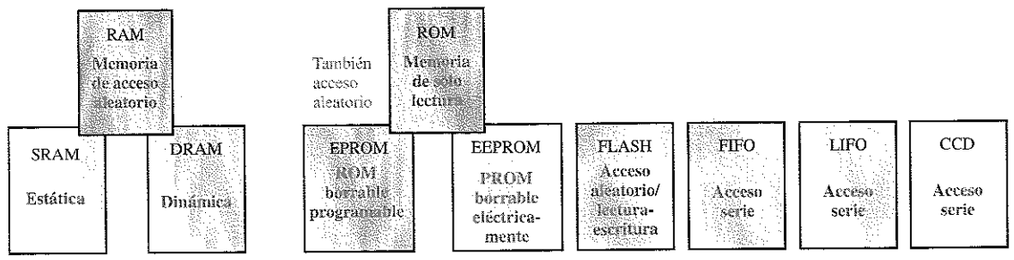
Teclado numérico

REVISIÓN DE LA SECCIÓN 12.10

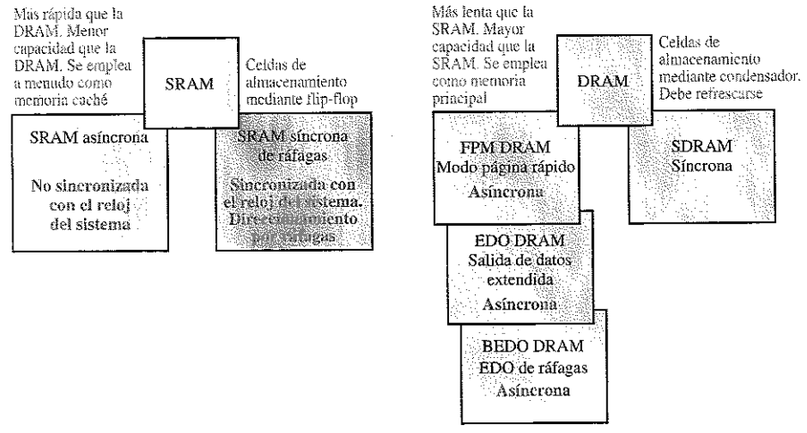
1. ¿Por qué tiene que reprogramarse el sistema con el código de entrada si se produce un fallo de alimentación y no se dispone de batería de reserva?
2. ¿Cuántas direcciones no se usan en la SRAM de la tarjeta de la lógica de memoria?
3. ¿Cómo inhibe el interruptor de modo de programación la lógica del código de seguridad durante el modo de programación?

■ RESUMEN

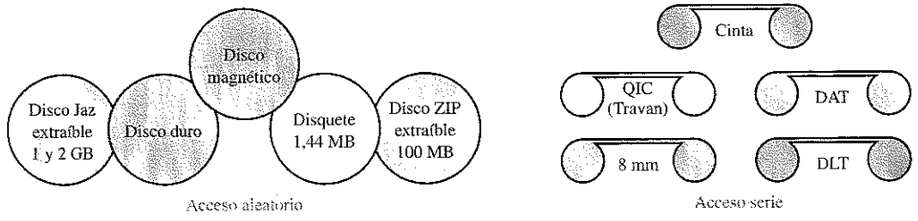
■ Tipos de memorias semiconductoras:



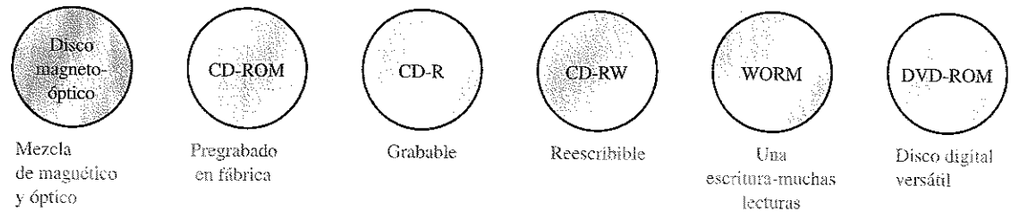
■ Tipos de memorias SRAM (RAM estática) y DRAM (RAM dinámica):



■ Tipos de dispositivos de almacenamiento magnético



■ Tipos de dispositivos de almacenamiento óptico (láser):



■ PALABRAS CLAVE

Estos términos también se encuentran en el glosario del final del libro.

BEDO DRAM Memoria de acceso aleatorio dinámica con salida de datos extendida a ráfagas.

Bus Grupo de líneas digitales que establece la interfaz entre uno o más dispositivos basándose en una especificación estandarizada.

Byte Grupo de ocho bits.

Capacidad Número total de unidades de datos (bits, nibbles, bytes, palabras) que puede almacenar una memoria.

Capacidad de palabra El número de palabras que una memoria puede almacenar.

Cartucho Jaz Un dispositivo de almacenamiento. Discos duros encerrados en un cartucho de plástico rígido, con capacidades de almacenamiento de 1 Gbyte o 2 Gbytes.

CCD Charge-Coupled Device, dispositivo de acoplamiento de carga; un tipo de memoria semiconductora que almacena datos en forma de paquetes de carga y a la que se accede en serie.

CD-R CD grabable, dispositivo de almacenamiento de disco óptico en el que los datos se pueden almacenar una vez.

CD-ROM Dispositivo de almacenamiento en disco óptico en el que los datos se almacenan de forma permanente.

CD-RW CD-reescribible; disco óptico en el que se pueden escribir y sobreescribir muchas veces los datos.

Celda Un elemento de almacenamiento en una memoria.

DAT Cinta digital de audio. Un tipo de formato de cinta magnética.

DIMM Módulo de memoria de terminal doble.

Dirección Posición de una determinada celda de almacenamiento o grupo de celdas en memoria.

Disco duro Dispositivo de almacenamiento magnético; normalmente, una pila de dos o más discos rígidos encerrados en un compartimento sellado.

Disco magneto-óptico Dispositivo de almacenamiento que utiliza medios electromagnéticos y un haz láser para leer y escribir datos.

Disco Zip Un tipo de dispositivo de almacenamiento magnético. Disco flexible con una capacidad de 100 Mbytes contenido en un cartucho de plástico rígido de aproximadamente el tamaño de un disquete.

Disquete o disco flexible Dispositivo de almacenamiento magnético. Típicamente suele ser un disco flexible de 3,5 pulgadas, con una capacidad de almacenamiento de 1,44 Mbytes, ubicado en el interior de una funda de plástico rígido.

DLT Digital Linear Tape (cinta digital lineal); un tipo de formato de cinta magnética.

DRAM Dynamic Random-Access Memory, memoria dinámica de acceso aleatorio. Un tipo de memoria semiconductora que utiliza condensadores como elemento de almacenamiento y es una memoria de lectura/escritura volátil.

DVD-ROM Digital Versatil Disk-ROM (disco digital versátil ROM); también conocido como disco digital de vídeo-ROM. Un tipo de dispositivo de almacenamiento óptico en el que los datos se prealmacenan y que tiene una capacidad mucho mayor que un CD-ROM.

EDO DRAM Memoria de acceso aleatorio dinámica con salida de datos extendida.

EEPROM Electrically Erasable Programmable Read-Only Memory, memoria programable de sólo lectura eléctricamente borrable. Un tipo de memoria semiconductora.

EPROM Memoria de sólo lectura programable y borrable. Un tipo de memoria semiconductora.

FET Field-Effect Transistor, transistor de efecto de campo.

FIFO First In-First Out, memoria de tipo Cola. Primero en entrar-primero en salir.

FPM DRAM Memoria de acceso aleatorio dinámica con modo de página rápido.

Lectura El proceso de recuperar datos de una memoria.

LIFO Last In-First Out, memoria de tipo primero en entrar-último en salir. Pila de memoria.

Longitud de palabra El número de bits en una palabra.

Matriz de memoria Matriz formada por las celdas de memoria colocadas formando filas y columnas.

Memoria caché Memoria de alta velocidad, relativamente pequeña, que almacena las instrucciones o datos más recientemente utilizados procedentes de la memoria principal, más grande pero más lenta.

Memoria flash Memoria semiconductora de acceso aleatorio de lectura/escritura no volátil, en la que los datos se almacenan como carga en la puerta flotante de un determinado tipo de FET.

Memoria dinámica Memoria semiconductora que tiene celdas de almacenamiento capacitivas que tienden a perder los datos que llevan almacenados más de un cierto periodo de tiempo y, por lo tanto, tienen que ser refrescadas.

Memoria estática Memoria semiconductora volátil que utiliza flip-flops como celdas de almacenamiento y que es capaz de mantener los datos indefinidamente sin necesidad de refresco, en presencia de alimentación.

MOSFET Metal-Oxide Semiconductor Field Effect Transistor, transistor de efecto campo de semiconductor metal-óxido.

Nibble Un grupo de cuatro bits.

No volátil Término que describe la memoria que puede mantener los datos almacenados cuando se desconecta la alimentación.

Palabra Unidad completa de datos binarios.

Periodo de latencia El tiempo que tarda el sector deseado en colocarse bajo el cabezal, una vez que el cabezal está posicionado sobre la pista deseada de un disco duro magnético.

Pipeline Cuando se aplica a las memorias, una implementación que permite que se inicie una operación de lectura o de escritura antes de que la operación anterior se complete.

PROM Programmable Read-Only Memory (memoria de sólo lectura programable); un tipo de memoria semiconductora.

QIC Quarter-inch cassette; un tipo de cinta magnética.

RAM Random-Access Memory, memoria de acceso aleatorio; memorias semiconductoras volátiles de lectura/escritura.

Refresco Renovar los contenidos de una memoria dinámica, recargando las celdas de almacenamiento de tipo capacitivo.

ROM Read-Only Memory, memoria semiconductora no volátil de acceso aleatorio.

SDRAM Memoria de acceso aleatorio dinámica síncrona.

SIMM Single-In-line Memory Module, módulo de memoria de una sola fila de terminales.

SRAM Static Random Access Memory, memoria estática de acceso aleatorio; un tipo de memoria volátil semiconductora de lectura/escritura.

Tiempo de acceso Tiempo que transcurre desde que se aplica un código de dirección válido en las entradas de una memoria hasta que aparecen datos de salida válidos.

Tiempo de búsqueda Tiempo que tarda el cabezal de lectura-escritura de un disco duro en posicionarse sobre la pista deseada para realizar una operación de lectura.

UV EPROM ROM programable borrable por rayos ultravioleta.

Volátil Término que describe una memoria en la que se pierden los datos almacenados cuando se elimina la alimentación.

WORM Write Once-Read Many (una escritura, muchas lecturas); un tipo de dispositivo de almacenamiento óptico.

■ AUTOTEST

1. La capacidad de bits de una memoria que tiene 1024 direcciones y que puede almacenar 8 bits en cada dirección es

   (a) 1024 **(b)** 8192 **(c)** 8 **(d)** 4096

2. Una palabra de datos de 32 bits está formada por

   (a) 2 bytes **(b)** 4 nibbles **(c)** 4 bytes **(d)** 3 bytes y 1 nibble

3. Los datos en una memoria de acceso aleatorio (RAM) se almacenan durante

   (a) la operación de lectura **(b)** la operación de habilitación
   (c) la operación de escritura **(d)** la operación de direccionamiento

4. Los datos que se almacenan en una determinada dirección de una memoria de acceso aleatorio (RAM) se pierden cuando

   (a) se apaga la alimentación
   (b) se leen los datos de dicha dirección
   (c) se escriben nuevos datos en dicha dirección
   (d) las respuestas (a) y (c)

5. Una ROM es

   (a) una memoria no volátil **(b)** una memoria volátil
   (c) una memoria de lectura/escritura **(d)** una memoria organizada en bytes.

6. Una memoria con 256 direcciones tiene

   (a) 256 líneas de dirección **(b)** 6 líneas de dirección
   (c) 1 línea de dirección **(d)** 8 líneas de dirección

7. Una memoria organizada en bytes tiene

   (a) 1 línea de salida de datos **(b)** 4 líneas de salida de datos
   (c) 8 líneas de salida de datos **(d)** 16 líneas de salida de datos

8. La celda de almacenamiento en una SRAM es

   (a) un flip-flop **(b)** un condensador
   (c) un fusible **(d)** un punto magnético

9. Una DRAM debe ser

   (a) reemplazada periódicamente
   (b) refrescada periódicamente
   (c) habilitada siempre
   (d) programada antes de cada uso

10. Una memoria flash es

    (a) volátil
    (b) una memoria de sólo lectura
    (c) una memoria de lectura/escritura
    (d) no volátil
    (e) las respuestas (a) y (c)
    (f) las respuestas (c) y (d)

11. Disco duro, disquete, disco Zip y disco Jaz son todos ellos

    (a) dispositivos de almacenamiento magneto-óptico
    (b) dispositivos de almacenamiento semiconductores
    (c) dispositivos de almacenamiento magnéticos
    (d) dispositivos de almacenamiento ópticos.

12. Los dispositivos de almacenamiento óptico emplean

    (a) luz ultravioleta **(b)** campos electromagnéticos
    (c) acopladores ópticos **(d)** láseres.

■ PROBLEMAS

SECCIÓN 12.1 Principios de las memorias semiconductoras

1. Identificar la ROM y la RAM de la Figura 12.76.

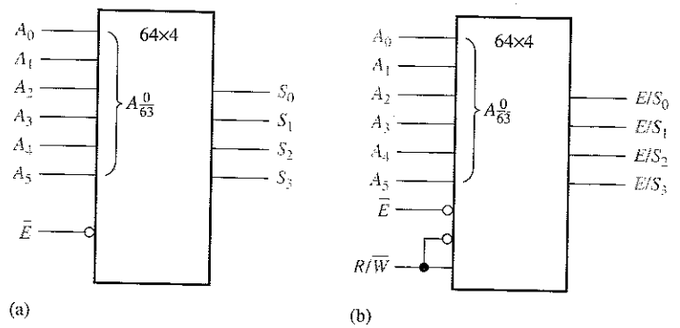


FIGURA 12.76

2. Explicar por qué las ROM y las RAM son memorias de acceso aleatorio.
3. Explicar los propósitos del bus de direcciones y del bus de datos.
4. Cuál es la dirección de memoria (de 0 hasta 256) que representa cada uno de los siguientes números hexadecimales:

   (a) $0A_{16}$ **(b)** $3F_{16}$ **(c)** CD_{16}

SECCIÓN 12.2 Memorias de acceso aleatorio (RAM)

5. En una matriz de memoria estática con cuatro filas similar a la de la Figura 12.8, se almacenan inicialmente todos ceros en todas las direcciones. ¿Cuál es el contenido después de las siguientes condiciones? Suponer que un 1 selecciona una fila.

   Fila 0 = 1, Entrada de datos (bit 0) = 1,
   Fila 1 = 0, Entrada de datos (bit 1) = 1,
   Fila 2 = 1, Entrada de datos (bit 2) = 1,
   Fila 3 = 0, Entrada de datos (bit 3) = 0

6. Dibujar un diagrama lógico básico para una RAM estática de 512 x 8 bits, indicando todas las entradas y salidas.
7. Suponiendo que una SRAM de 64k x 8 tiene una estructura similar a la SRAM de la Figura 12.10, determinar el número de filas y de columnas de 8 bits en la matriz de celdas de memoria.
8. Dibujar de nuevo el diagrama de bloques de la Figura 12.10 para una memoria de 64k x 8.
9. Explicar la diferencia entre una SRAM y una DRAM.
10. ¿Cuál es la capacidad de una DRAM con doce líneas de dirección?

SECCIÓN 12.3 Memorias de sólo lectura (ROM)

11. Para la matriz ROM de la Figura 12.77, determinar las salidas para todas las posibles combinaciones de entrada, y resumirlas en forma de tabla (celda en blanco es 1, celda oscura es 0).
12. Determinar la tabla de verdad de la ROM de la Figura 12.78.
13. Utilizando un procedimiento similar al del Ejemplo 12-1, diseñar una ROM para convertir un único dígito BCD en código de exceso 3.

14. ¿Cuál es la capacidad total de bits de una ROM que tiene 14 líneas de dirección y 8 salidas de datos?

FIGURA 12.77

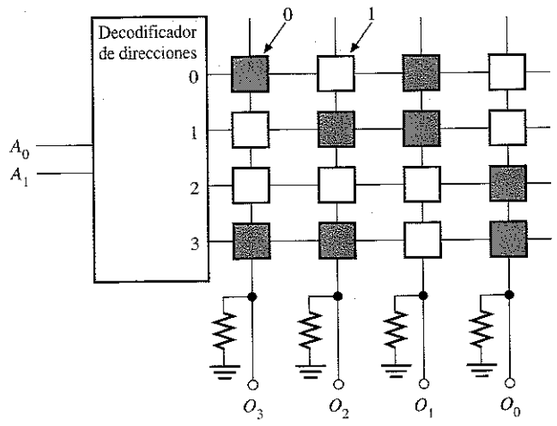
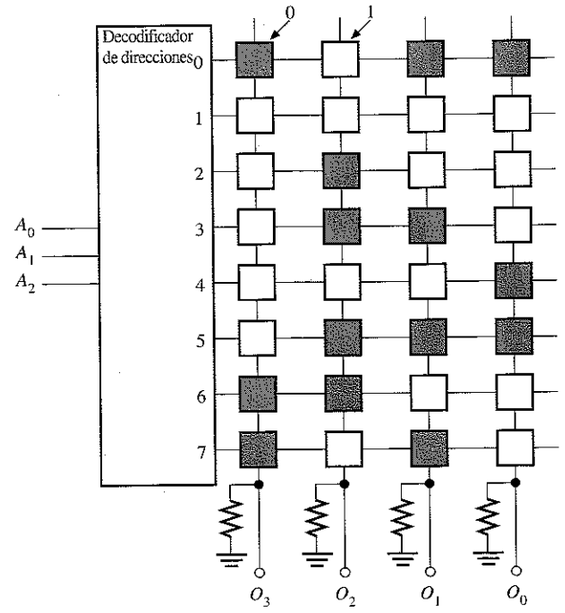


FIGURA 12.78



SECCIÓN 12.4 Memorias ROM programables (PROM y EPROM)

15. Suponer que la matriz PROM de la Figura 12.79 se programa fundiendo un hilo fusible para crear un 0. Indicar los hilos que hay que fundir para programar una tabla índice para la operación X^3, donde X es un número de 0 a 7.

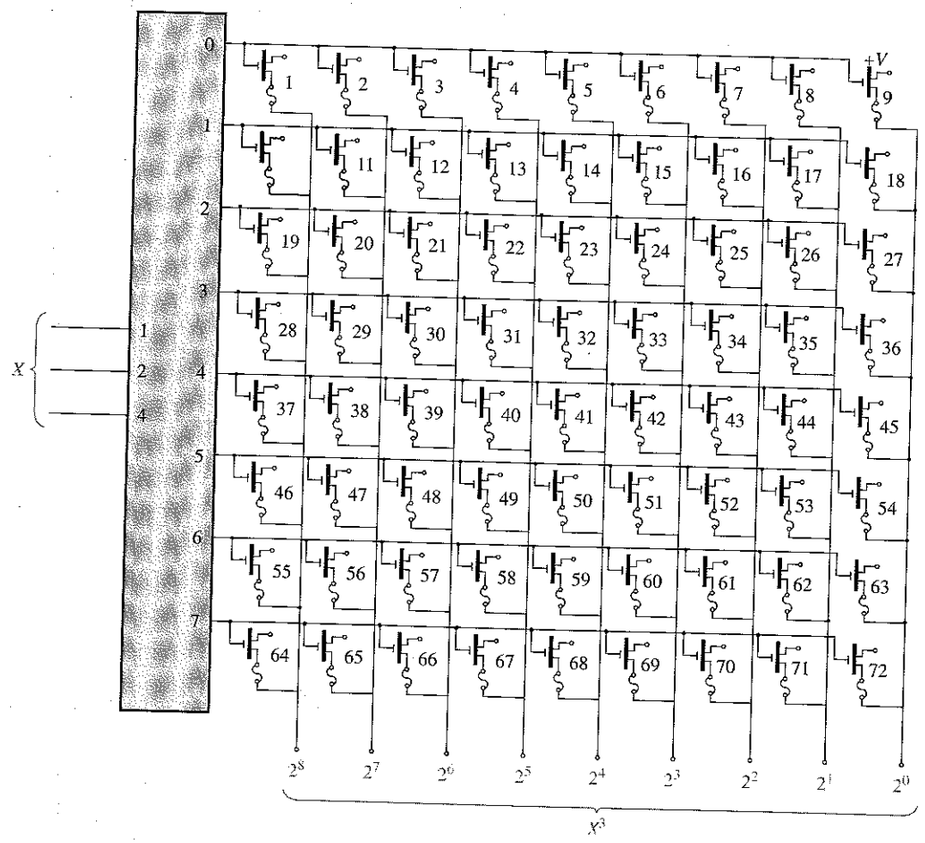


FIGURA 12.79

16. Determinar las direcciones que se programan y los contenidos de cada dirección después de aplicar la secuencia de programación de la Figura 12.80 a una PROM como la mostrada en la Figura 12.31.

SECCIÓN 12.6 Expansión de memoria

17. Utilizar memorias DRAM de 16k x 4 par formar una RAM de 64k x 8. Dibujar el diagrama lógico.

18. Utilizando un diagrama de bloques, demostrar cómo se pueden expandir memorias RAM dinámicas de 64k x 1 para formar una RAM de 256k x 4.

19. ¿Cuál es la longitud de palabra y la capacidad de palabra de la memoria del problema 17? ¿Y del problema 18?

SECCIÓN 12.7 Tipos especiales de memorias

20. Completar el cronograma de la Figura 12.81, mostrando las formas de onda de salida, que inicialmente están a nivel BAJO, para una memoria serie FIFO como la mostrada en la Figura 12.49.

21. Considerar una RAM de 4096 x 8 en la que las 64 últimas direcciones se usan como pila LIFO. Si la primera dirección de la RAM es 000_{16}, indicar las 64 direcciones utilizadas para la pila.

22. En la memoria del problema 21, se introducen 16 bytes en la pila. ¿En qué dirección se encuentra el primer byte? ¿Y el último byte?

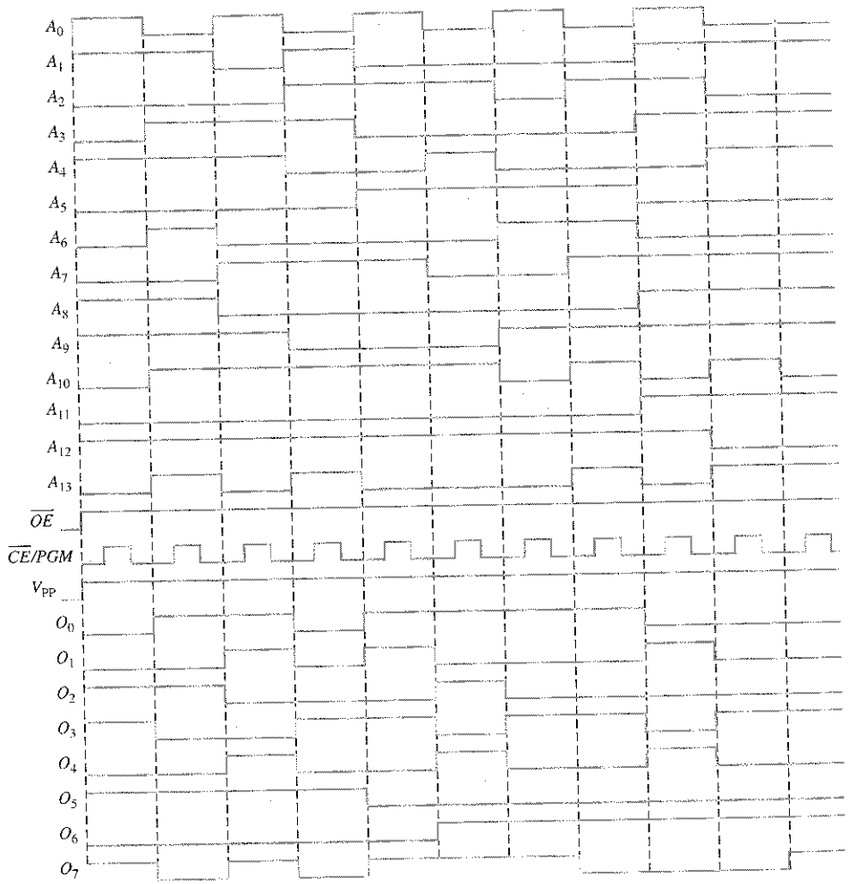


FIGURA 12.80

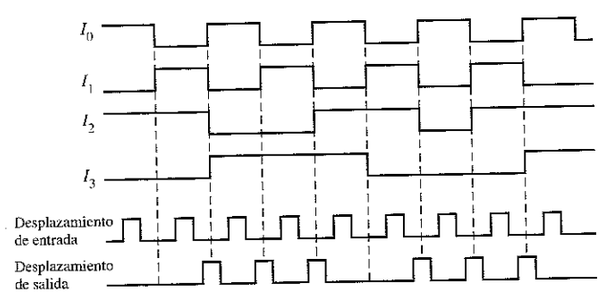


FIGURA 12.81

SECCIÓN 12.8 Dispositivos de almacenamiento magnético y óptico

23. Describir el formato general de un disco duro.

24. Explicar qué es el tiempo de búsqueda y el periodo de latencia en una unidad de disco duro.

25. ¿Por qué la cinta magnética requiere un tiempo de acceso mucho mayor que un disco?

26. Explicar las diferencias entre un disco magneto-óptico, un CD-ROM y un WORM.

SECCIÓN 12.9 Pruebas y localización de averías

27. Determinar si los contenidos de la ROM de la Figura 12.82 son correctos.

FIGURA 12.82

ROM	
1 0 1 1 1	
1 1 1 1 0	
1 1 0 1 1	
1 0 1 1 0	
1 1 1 0 1	
1 1 1 0 0	
0 0 0 0 1	
0 1 1 0 0	Suma de comprobación

28. Una ROM de 128k x 8 se implementa como se indica en la Figura 12.83. El decodificador decodifica los dos bits de dirección más significativos para habilitar una de las ROM cada vez, dependiendo de la dirección seleccionada.

(a) Expresar la dirección más baja y la más alta de cada ROM con números hexadecimales.

(b) Suponer que se usa una única suma de comprobación para la memoria completa y se almacena en la dirección más alta. Desarrollar el organigrama para probar el sistema de memoria completo.

(c) Suponer que cada ROM tiene una suma de comprobación almacenada en su dirección más alta. Modificar el organigrama desarrollado en el apartado (b) para reflejar este cambio.

(d) ¿Cuál es la desventaja de utilizar una única suma de comprobación para la memoria completa en lugar de una suma de comprobación para cada ROM individual?

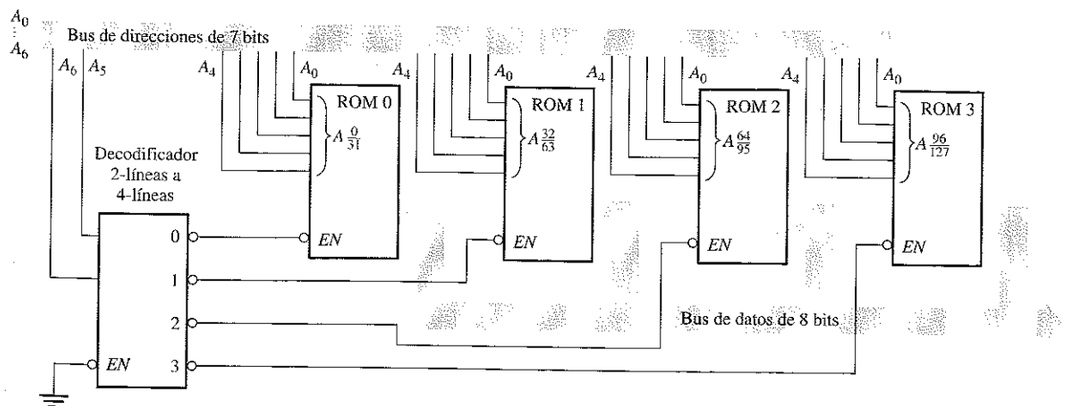


FIGURA 12.83

29. Suponer que se ejecuta una prueba de suma de comprobación en la memoria de la Figura 12.83 y que cada ROM tiene su suma de comprobación en la dirección más alta. ¿Cuál o cuáles circuitos integrados se reemplazarían para cada uno de los siguientes mensajes de error que aparecen en el monitor de vídeo del sistema?

(a) DIRECCIÓN 40 - 5F FALLO
(b) DIRECCIÓN 20 - 3F FALLO
(c) DIRECCIÓN 00 - 7F FALLO

SECCIÓN 12.10 Aplicación a los sistemas digitales

30. Desarrollar un cronograma de la lógica de memoria básica de la Figura 12.74, para ilustrar la introducción de los dígitos 4321 en la RAM. Incluir todas las entradas y salidas de los dispositivos.

31. En el modo de programación, ¿cuál es el estado del contador de la Figura 12.74 después de haber introducido dos dígitos del código?

32. ¿Cuál es el propósito del 74LS27 de la tarjeta de la lógica de memoria en la Práctica de sistemas 2?

33. Comentar las ventajas y desventajas de utilizar una PROM en lugar de una SRAM en la lógica de memoria.

Problemas especiales de diseño

34. Desarrollar un diagrama de bloques básico para el comprobador de ROM de la Figura 12.67, para probar la ROM de la Figura 12.84.

35. Repetir el problema 34 para la ROM de la Figura 12.85.

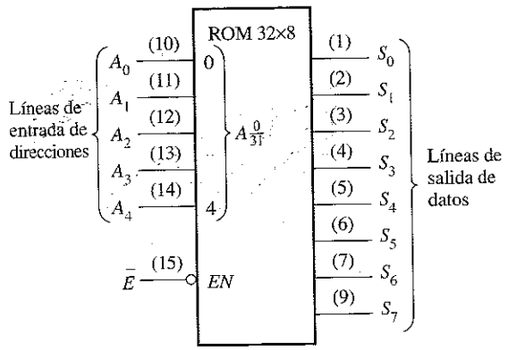


FIGURA 12.84

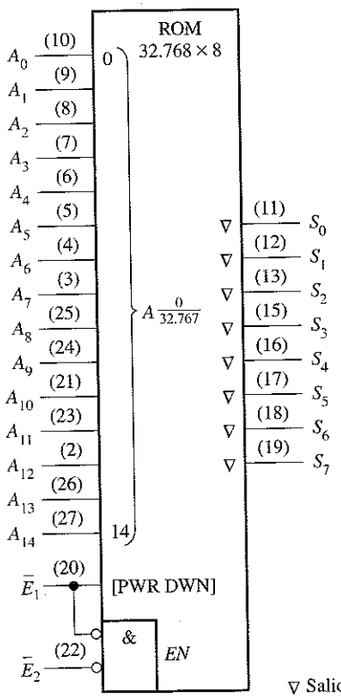


FIGURA 12.85

▽ Salida triestado

36. Modificar, si es necesario, el diseño de la lógica de memoria del sistema de seguridad de acceso, para acomodar un código de acceso de 5 dígitos.

37. Realizar las modificaciones apropiadas en la lógica de introducción del código en el sistema de seguridad de acceso, para un código de acceso de 5 dígitos. Consultar el Capítulo 10.

■ RESPUESTAS A LAS REVISIONES DE CADA SECCIÓN

SECCIÓN 12.1

1. El bit es la unidad de datos más pequeña.
2. 2048 bits son 256 bytes.
3. Una operación de escritura almacena datos en la memoria.
4. Una operación de lectura realiza una copia de datos de la memoria.
5. Una unidad de datos se localiza mediante una dirección.
6. Una RAM es volátil y tiene capacidad de lectura/escritura. Una ROM es no volátil y sólo tiene capacidad de lectura.

SECCIÓN 12.2

1. Ráfaga síncrona y asíncrona.
2. Una pequeña memoria rápida entre la UCP y la memoria principal.
3. Las SRAM tienen celdas de almacenamiento latch que pueden mantener indefinidamente los datos. Las DRAM tienen celdas de almacenamiento capacitivo que se deben refrescar periódicamente.
4. La operación de refresco evita que los datos se pierdan debido a la descarga del condensador. Cada bit almacenado se restaura periódicamente recargando el condensador.
5. FPM, EDO, BEDO, síncrona

SECCIÓN 12.3

1. 512×8 es igual a 4096 bits.
2. ROM de máscara, PROM, EPROM, UV EPROM, EEPROM.
3. Se requieren ocho bits de dirección para 256 localizaciones de bytes ($2^8 = 256$).

SECCIÓN 12.4

1. Las PROM son programables eléctricamente, las ROM no.
2. Los bits quedan a 1 después de borrar la EPROM.
3. La lectura es el modo normal de operación de una PROM.

SECCIÓN 12.5

1. Flash, ROM, EPROM y EEPROM son no volátiles.
2. La memoria flash es no volátil. Las memorias SRAM y DRAM son volátiles.
3. Programación, lectura, borrado.

SECCIÓN 12.6

1. Ocho memorias RAM.
2. Ocho memorias RAM.

3. SIMM: módulo de memoria de una única fila de pines.
4. DIMM: módulo de memoria con doble fila de pines.

SECCIÓN 12.7

1. En una memoria FIFO el *primer* bit (o palabra) *en entrar* es el *primer* bit (o palabra) *en salir.*
2. En una memoria LIFO el *último* bit (o palabra) *en entrar* es el *primer* bit (o palabra) *en salir.* Una pila es una LIFO.
3. La operación o instrucción que añade datos a la pila de memoria.
4. La operación o instrucción que elimina datos de la pila de memoria.
5. CCD significa dispositivo de acoplamiento de carga.

SECCIÓN 12.8

1. Almacenamiento magnético: disquetes, disco duro, cinta y disco magneto-óptico.
2. Capacidad de almacenamiento de los discos flexibles: 1,44 Mbytes.
3. Un disco magnético está organizado en pistas y sectores.
4. Un disco magneto-óptico utiliza un rayo láser y un electroimán.
5. Zip: 100 Mbytes y Jaz:1 Gbyte y 2 Gbytes.
6. Almacenamiento óptico: CD-ROM, CD-R, CD-RW, DVD-ROM, WORM.

SECCIÓN 12.9

1. Los contenidos de la ROM se suman y comparan con una suma de comprobación prealmacenada.
2. La suma de comprobación no se puede utilizar porque los contenidos de una RAM no son fijos.
3. (1) un cortocircuito entre celdas adyacentes; (2) la incapacidad de algunas celdas para almacenar tanto 1s como 0s; (3) la alteración dinámica de los contenidos de una dirección cuando los contenidos de otra varían.

SECCIÓN 12.10

1. El sistema debe reprogramarse porque la SRAM es volátil y los datos almacenados se pierden cuando se desconecta la alimentación.
2. Se usan cinco de las dieciséis direcciones. Las restantes once direcciones no se emplean.
3. El interruptor de modo de programación aplica un nivel BAJO a la puerta negativa-OR (NAND) que dispara el monoestable A en la lógica del código de seguridad.

■ RESPUESTAS A LOS PROBLEMAS RELACIONADOS DE LOS EJEMPLOS

12.1 $G_3G_2G_1G_0 = 1110$

12.2 Deben conectarse ocho memorias ROM de 64k x1 en paralelo para formar una ROM de 64k × 8.

12.3 Dieciséis memorias ROM de 64k x 1.

12.4 Véase la Figura 12.86.

12.5 ROM 1: 0 a 524.287; ROM 2: 524.288 a 1.048.575

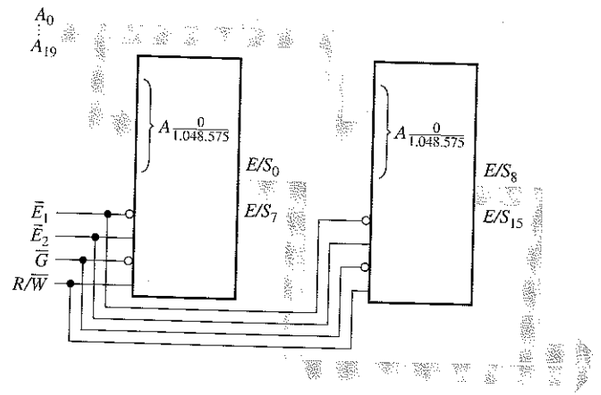


FIGURA 12.86

■ RESPUESTAS AL AUTOTEST

1. (b) **2.** (c) **3.** (c) **4.** (d) **5.** (a) **6.** (d) **7.** (c) **8.** (a)
9. (b) **10.** (f) **11.** (c) **12.** (d)

13

INTERFACES

13.1 Interfaces entre el mundo digital y el analógico
13.2 Conversión digital-analógica (D/A)
13.3 Conversión analógica-digital (A/D)
13.4 Localización de averías en los convertidores DAC y ADC
13.5 Interfaces internas de los sistemas
13.6 Buses estándar
13.7 Aplicación a los sistemas digitales

Simulación por computadora de circuitos digitales:
El tutorial de Electronics Workbench (EWB) se encuentra en la dirección http://www.prenhall.com/floyd.

■ OBJETIVOS DEL CAPÍTULO

- ❑ Describir aplicaciones de las conversiones digital-analógica y analógica-digital.
- ❑ Explicar la conversión D/A mediante el método de ponderación binaria y mediante el método de la red en escalera *R/2R*.
- ❑ Explicar la conversión A/D mediante cualquiera de los siguientes procesos: paralelo (flash), rampa en escalera, seguimiento, pendiente simple, pendiente doble y aproximaciones sucesivas.
- ❑ Reconocer los errores de conversión, como la no monotonicidad, la no linealidad diferencial, el offset, la ganancia inadecuada y los códigos incorrectos o ausentes en los convertidores analógico-digital (ADC) y digital-analógico (DAC).
- ❑ Localizar averías en los DAC y ADC.
- ❑ Explicar cómo implementar la interfaz con los dispositivos digitales utilizando sistemas de buses.
- ❑ Definir las características y aplicaciones básicas de los buses estándar internos PCI e ISA.
- ❑ Definir las características y aplicaciones básicas de los buses estándar externos RS-232C, IEEE 1394 (FireWire), USB, IEEE 488 (GPIB) y SCSI.

■ INTRODUCCIÓN

Implementar una **interfaz** es el proceso de hacer que dos o más dispositivos o sistemas sean operacionalmente compatibles entre sí, de modo que puedan trabajar juntos del modo que se requiera.

Muchas magnitudes son de naturaleza analógica, es decir, son magnitudes continuas. Las magnitudes físicas tales como temperatura, presión, tiempo y velocidad son ejemplos de magnitudes analógicas. En este capítulo se introducen los métodos de conversión de códigos digitales a magnitudes analógicas. Esto se denomina conversión digital-analógica (conversión D/A). También se cubren métodos para llevar a cabo la conversión de magnitudes analógicas en códigos digitales. Esto se denomina conversión analógico-digital (conversión A/D).

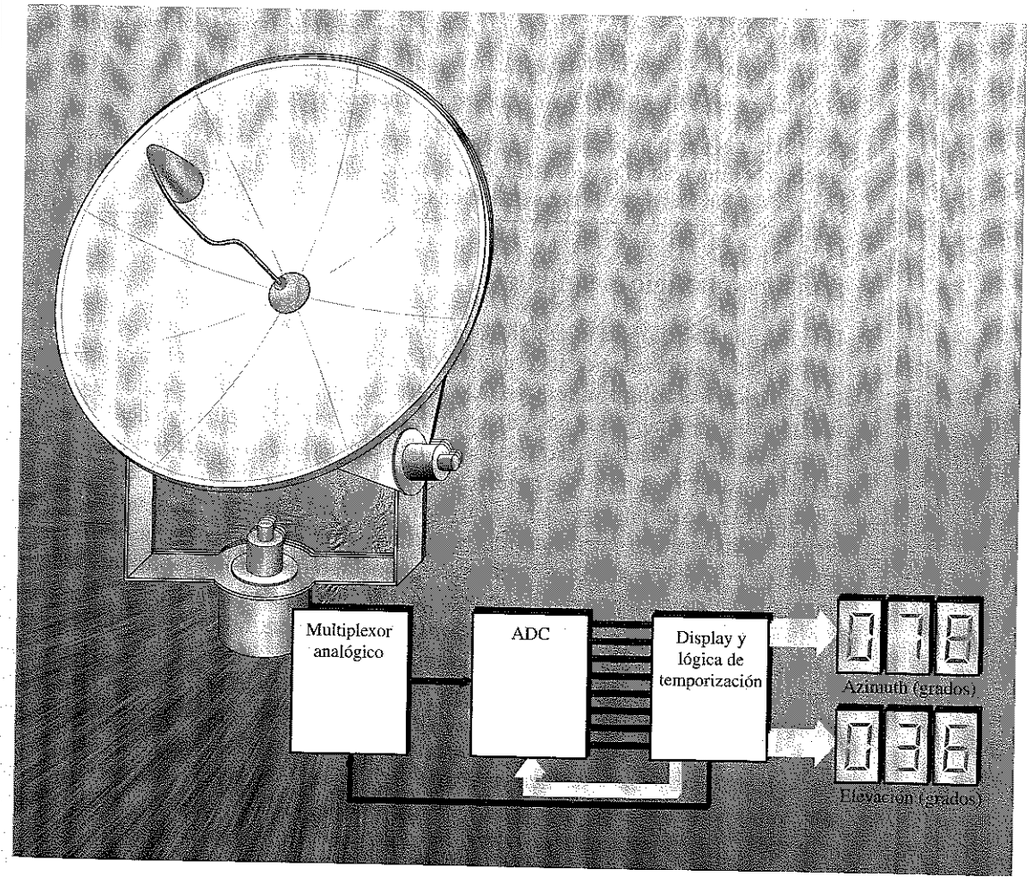
Además de las conversiones D/A y A/D, se presentan varios tipos de buses estándar.

DISPOSITIVOS ESPECÍFICOS

DAC0808 ADC0804 AD9300
AD673

■ APLICACIÓN A LOS SISTEMAS DIGITALES

Esta aplicación a un sistema digital ilustra los conceptos que se aprenderán en el capítulo, y se centra tanto en la parte analógica como digital de un sistema de posicionamiento de antena. Se requiere una conversión A/D para convertir las tensiones analógicas que representan la posición de la antena a formato digital, y también se precisan circuitos lógicos para procesar y presentar en el display las posiciones angulares.



13.1 ■ INTERFACES ENTRE EL MUNDO DIGITAL Y EL ANALÓGICO

En el Capítulo 1 se han descrito brevemente las magnitudes analógicas y digitales y se apuntaron las diferencias entre ellas. Algunas veces, las magnitudes analógicas se denominan magnitudes del mundo real, ya que la mayoría de las magnitudes físicas que resulta interesante medir son analógicas por naturaleza. Muchas aplicaciones de computadoras y otros sistemas digitales requieren como entrada magnitudes del mundo real, tales como la temperatura, velocidad, posición, presión y fuerza. Las magnitudes del mundo real pueden incluso abarcar imágenes gráficas. También, frecuentemente, los sistemas digitales deben generar salidas para controlar las magnitudes del mundo real.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Definir una magnitud analógica. ■ Enunciar el propósito de la conversión analógico-digital. ■ Enunciar el propósito de la conversión digital-analógica. ■ Proporcionar ejemplos de ambos tipos de conversión, analógico-digital y digital-analógica.

Señales analógicas y digitales

Una magnitud analógica es aquella que tiene valores continuos dentro de un determinado rango, a diferencia de un conjunto de valores discretos para el caso de las señales digitales. Para ilustrar la diferencia entre la representación analógica y digital de una magnitud, tomemos el caso de una tensión que varía en el rango de 0 V a +15 V. La representación analógica de esta magnitud toma *todos* los valores entre 0 y +15 V, lo que constituye un rango infinito de valores.

En el caso de una representación digital de la tensión empleando un código binario de 4 bits, sólo se pueden definir dieciséis valores. Para representar más valores entre 0 y +15 deben utilizarse más bits en el código digital. Por tanto, una magnitud analógica se puede representar con un cierto grado de precisión utilizando un código digital, que representa valores discretos específicos dentro del rango. En la Figura 13.1 se ilustra este concepto, donde la función analógica mostrada es una curva que varía suavemente tomando valores entre 0 y +15 V. Si se usa un código de 4 bits para representar esta curva, cada número binario representa un punto discreto de la curva.

La tensión representada sobre la curva de la Figura 13.1 se mide o muestrea en cada uno de los treinta y cinco intervalos iguales. La tensión en cada uno de estos intervalos se representa mediante un código de 4 bits, tal y como se indica. En esta situación, se tiene una serie de números binarios que representan distintos valores de tensión a lo largo de la curva analógica. Ésta es la idea fundamental de la conversión analógico-digital (A/D).

Se puede reconstruir una aproximación de la función analógica de la Figura 13.1 por medio de la secuencia de números digitales que se ha generado. Obviamente, habrá algunos errores en esta reconstrucción, ya que sólo se representan ciertos valores (en este ejemplo, treinta y seis) y no el conjunto de valores continuos. Si todos los valores digitales de los treinta y seis puntos se llevan a la gráfica de la Figura 13.2, se obtiene la función reconstruida. Como puede verse, la gráfica sólo se aproxima a la curva original, dado que los valores entre los puntos se desconocen. Se puede aumentar la precisión muestreando más frecuentemente la curva y aumentando el número de bits utilizados para representar el valor de cada muestra.

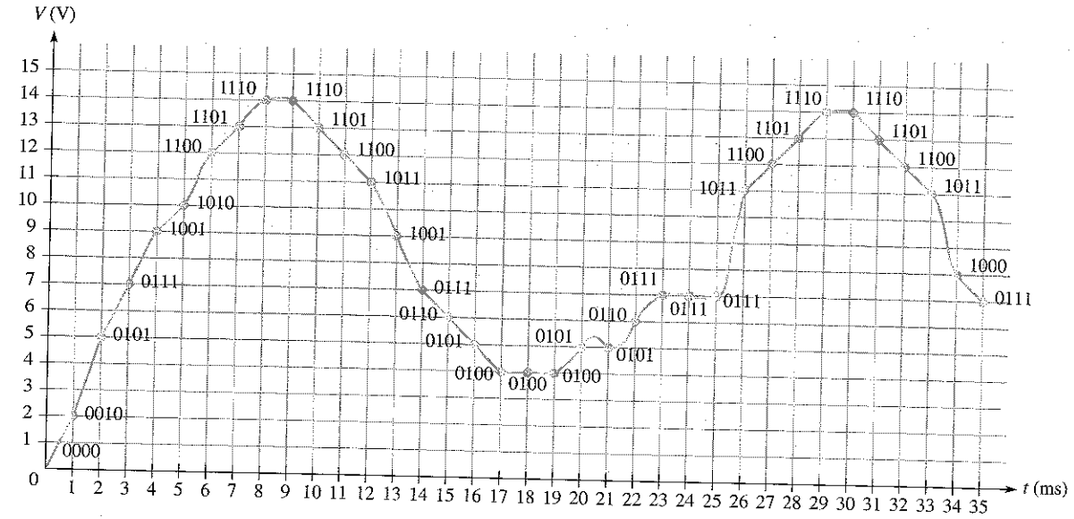


FIGURA 13.1
Puntos discretos (digitales) sobre una curva analógica.

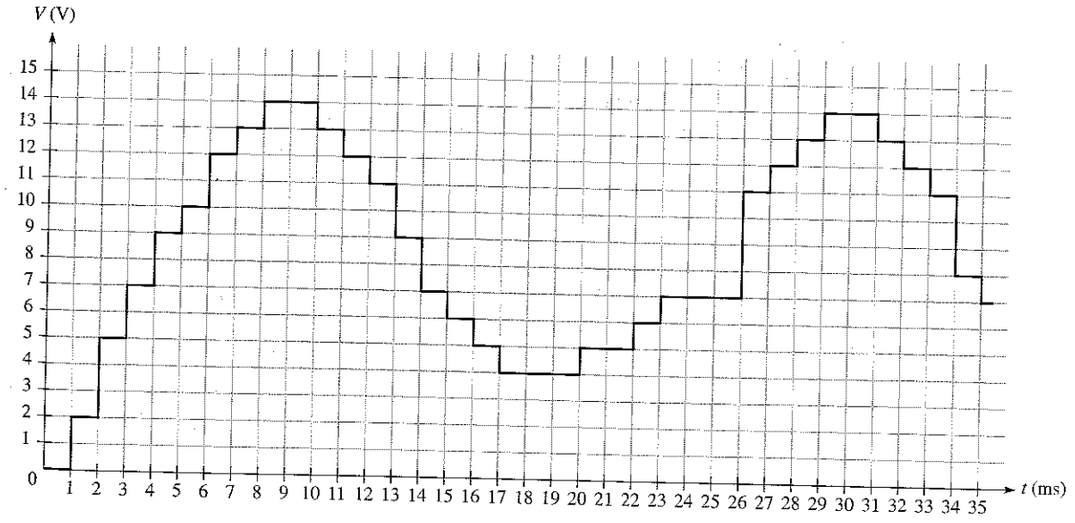


FIGURA 13.2
Reproducción digital aproximada de la curva analógica de la Figura 13.1.

Ejemplos de aplicación

Para realizar la interfaz entre los mundos analógico y digital se requieren dos procesos básicos que son: la **conversión analógico-digital (A/D)** y la **conversión digital-analógica (D/A)**. Los siguientes tres ejemplos de sistemas ilustran las aplicaciones de estos procesos de conversión.

Termostato electrónico. En la Figura 13.3 se presenta el diagrama de bloques simplificado de un termostato electrónico. El sensor de temperatura de la habitación produce una tensión analógica proporcional a la temperatura. Esta tensión se incrementa mediante el amplificador lineal y se aplica al **convertidor analógico-digital** (**ADC**, analog-to-digital converter), en el que se convierte en un código digital y es periódicamente muestreada por la lógica de control. Por ejemplo, supongamos que la temperatura de la habitación es 20 °C. A esta temperatura le corresponderá un valor de tensión concreto que aparecerá en la entrada del ADC y que será convertido en el número binario de 8 bits 00010100.

La lógica de control compara este número binario con un número binario que representa la temperatura deseada (es decir, 00010110 para 22 °C). Este valor deseado se ha introducido previamente a través del teclado numérico y se ha almacenado en un registro. La comparación indica que la temperatura real de la habitación es menor que la temperatura deseada. Como resultado de ello, la lógica de control manda al circuito de control de la unidad encender el calefactor. Mientras funciona el calefactor, la lógica de control continúa supervisando la temperatura real por mèdio del ADC. Cuando la temperatura real iguala o excede a la temperatura deseada, la lógica de control apaga el calefactor. Ambas temperaturas, real y deseada, se visualizan en un display.

Reproductor de CD. El reproductor de discos compactos es un ejemplo de un sistema que utiliza un **convertidor digital-analógico** (DAC, digital-to-analog converter). En la Figura 13.4 se presenta un diagrama de bloques básico. Una señal de audio se graba digitalmente en el CD, mediante una serie de muescas, que son detectadas por el cabezal láser y amplificadas. La señal digital amplificada se convierte en una secuencia de códigos binarios que representan la señal de audio originalmente grabada. La representación binaria de la señal de audio se convierte a formato analógico mediante el DAC, se amplifica y se envía a los altavoces.

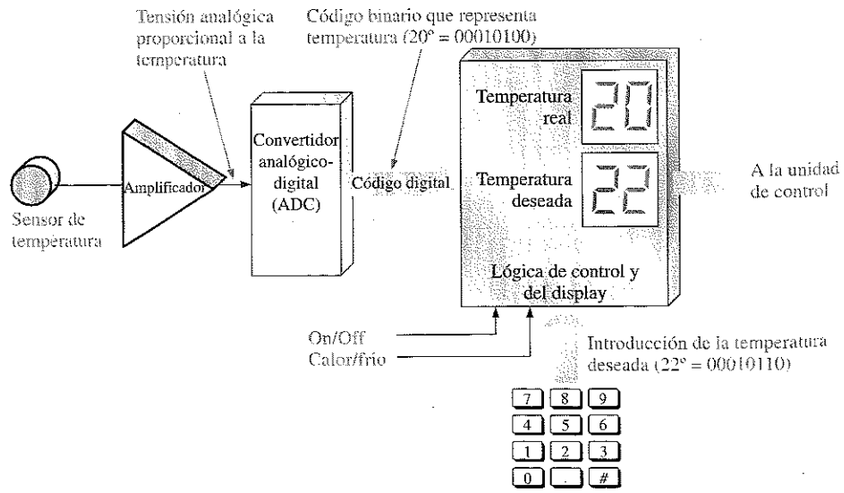


FIGURA 13.3
Diagrama de bloques básico de un termostato electrónico que utiliza un ADC.

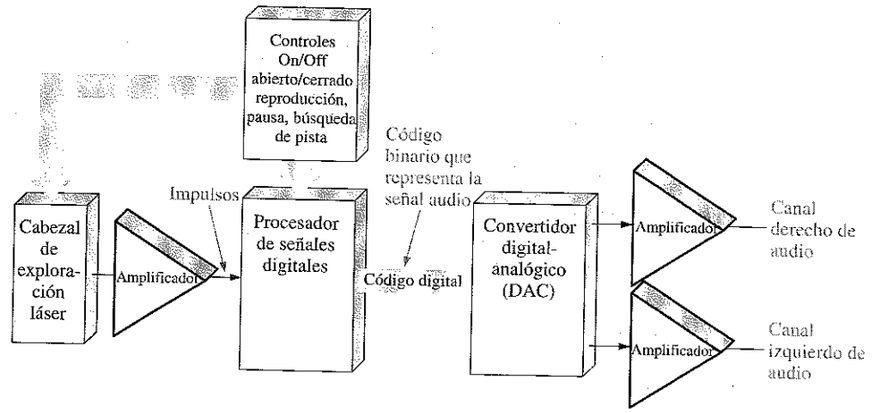


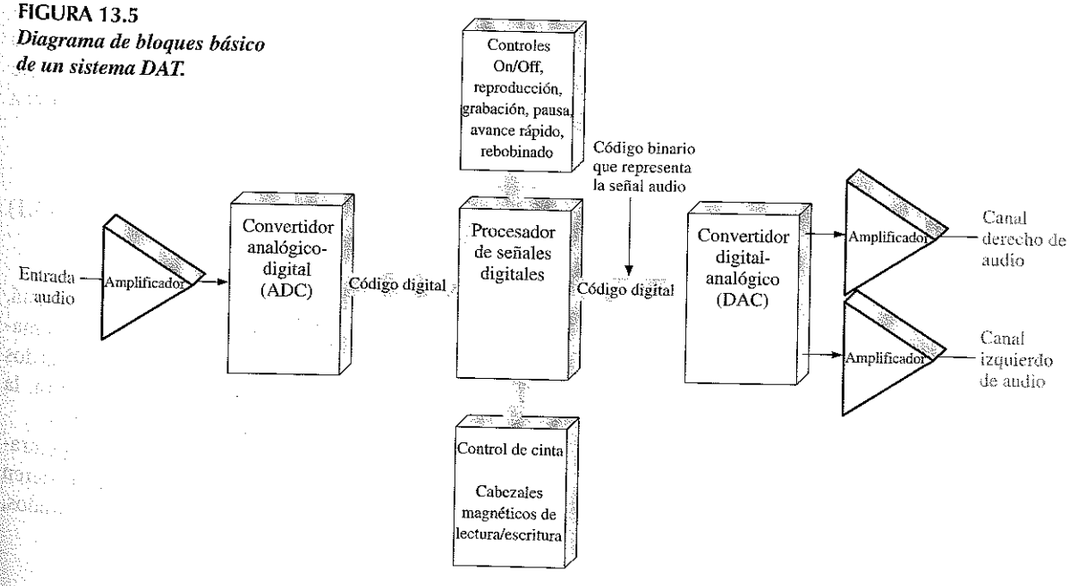
FIGURA 13.4
Diagrama de bloques básico de un reproductor de CD.

Grabador/reproductor digital de cintas de audio (DAT, Digital Audiotape). Otro sistema ejemplo que incluye las conversiones A/D y D/A es el grabador/reproductor **DAT**. En la Figura 13.5 se presenta su diagrama de bloques básico.

Por supuesto, una señal de audio es una magnitud analógica. En el modo de grabación, el sonido se capta, se amplifica y se convierte a formato digital mediante el ADC. Los códigos digitales que representan la señal de audio se procesan y registran en la cinta.

En el modo de reproducción, la señal de audio digitalizada se lee de la cinta, se procesa y se convierte de nuevo a formato analógico mediante el DAC. Luego se amplifica y se envía al sistema de altavoces.

FIGURA 13.5
Diagrama de bloques básico de un sistema DAT.



REVISIÓN DE LA SECCIÓN 13.1

1. En la naturaleza ¿en qué formato se presentan normalmente las magnitudes?
2. Explicar el propósito fundamental de la conversión A/D.
3. Explicar el propósito fundamental de la conversión D/A.

13.2 ■ CONVERSIÓN DIGITAL-ANALÓGICA (D/A)

La conversión D/A es una parte muy importante en muchos sistemas. En esta sección, examinaremos la teoría de funcionamiento de dos tipos básicos de convertidores digital-analógicos (DAC) y se aprenderán sus características de funcionamiento.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar básicamente qué es un amplificador operacional. ■ Mostrar cómo el amplificador operacional se puede usar como amplificador inversor o como comparador. ■ Explicar el funcionamiento de un DAC con ponderación binaria. ■ Explicar el funcionamiento de un DAC en escalera *R/2R*. ■ Describir los conceptos de resolución, precisión, linealidad, monotonicidad y tiempo de asentamiento de un DAC.

El amplificador operacional

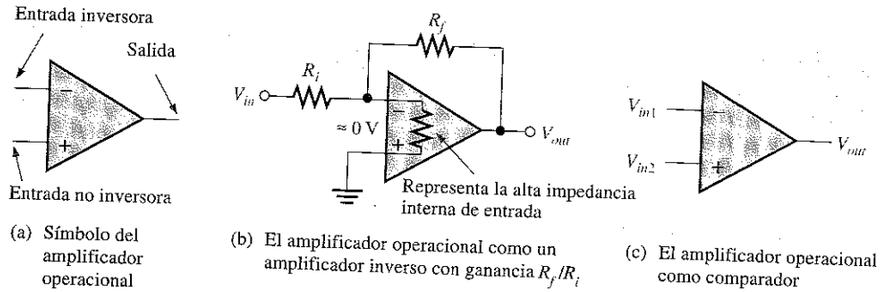
Antes de abordar los convertidores digital-analógicos (DAC), veremos brevemente un elemento muy común en la mayoría de los DAC y de los convertidores analógico-digitales (ADC). Este elemento es el amplificador operacional (AO).

Un **amplificador operacional** es un amplificador lineal que tiene dos entradas (inversora y no inversora) y una salida. Tiene una alta ganancia en tensión y una muy alta impedancia de entrada, así como una muy baja impedancia de salida. En la Figura 13.6(a) se muestra el símbolo del AO. Cuando se utiliza como amplificador inversor, se configura como se indica en la parte (b) de la figura. La resistencia de realimentación, R_f, y la resistencia de entrada, R_i, controlan la ganancia en tensión de acuerdo con la fórmula de la Ecuación 13.1, donde V_{out} / V_{in} es la ganancia en tensión en bucle cerrado (bucle cerrado se refiere a la realimentación desde la salida hasta la entrada por medio de R_f). El signo negativo indica inversión.

$$\frac{V_{out}}{V_{in}} = -\frac{R_f}{R_i} \qquad \textbf{(13.1)}$$

En la configuración de amplificador inversor, la entrada inversora del AO está, aproximadamente, al potencial de tierra (0 V), porque la realimentación y la extremadamente alta ganancia en bucle abierto hacen que la diferencia de tensión entre las dos entradas sea muy pequeña. Por tanto, como la entrada no inversora está a tierra, la entrada inversora está aproximadamente a 0 V, lo que se denomina *tierra virtual.*

Cuando el AO se utiliza como comparador se aplican dos tensiones a las entradas, como se muestra en la Figura 13.6(c). Cuando estas tensiones de entrada difieren en una pequeña cantidad, el AO pasa a uno de sus dos estados de salida saturados, nivel ALTO o BAJO, dependiendo de qué tensión sea mayor.



(a) Símbolo del amplificador operacional

(b) El amplificador operacional como un amplificador inverso con ganancia R_f/R_i

(c) El amplificador operacional como comparador

FIGURA 13.6
El amplificador operacional.

Convertidor analógico-digital con ponderación binaria

Existe un método de conversión D/A que utiliza una red resistiva en la que los valores de las resistencias representan los pesos binarios de los bits de entrada del código digital. La Figura 13.7 muestra un DAC de 4 bits de este tipo. Por cada una de las resistencias de entrada puede circular o no corriente, dependiendo del nivel de tensión de entrada. Si la tensión de entrada es cero (0 binario), la corriente también es cero. Si la tensión de entrada es un nivel ALTO (1 binario), la cantidad de corriente depende del valor de la resistencia de entrada y es diferente para cada una de las resistencias, como se indica en la figura.

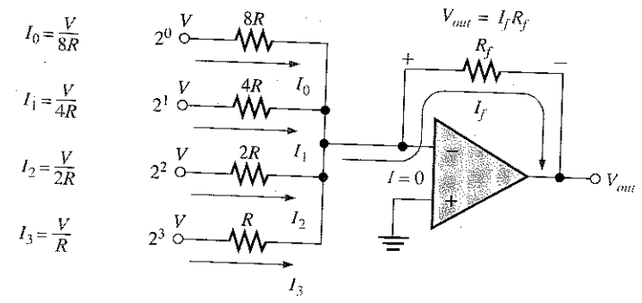


FIGURA 13.7
DAC con ponderación binaria de 4 bits.

Puesto que, prácticamente, no circula corriente por la entrada inversora (–) del AO, la suma de todas las corrientes de entrada pasa a través de R_f. Como la entrada inversora está a 0 V (tierra virtual), la caída en R_f es igual a la tensión de salida, es decir, $V_{out} = I_f R_f$.

Los valores de las resistencias de entrada se seleccionan de modo que sean inversamente proporcionales a los pesos binarios de los correspondientes bits de entrada. La resistencia de menor valor (R) corresponde a la entrada ponderada más

alta (2^3). Las restantes resistencias son múltiplos de R ($2R$, $4R$ y $8R$) y corresponden a los pesos binarios 2^2, 2^1 y 2^0, respectivamente. Las corrientes de entrada también son proporcionales a los pesos binarios. Luego la tensión de salida es proporcional a la suma de los pesos binarios, ya que es la suma de las corrientes de entrada por R_f.

Una de las desventajas de este tipo de DAC es el número de resistencias diferentes que utiliza. Por ejemplo, un convertidor de 8 bits requiere ocho resistencias en el rango que va desde R hasta $128R$, en pasos ponderados. Este rango de resistencias requiere tolerancias de 1 entre 255 (menor del 0,5%) para convertir la entrada con precisión, lo que hace muy difícil fabricar este tipo de DAC en grandes cantidades.

EJEMPLO 13.1 Determinar la salida del DAC de la Figura 13.8(a), si se aplican a las entradas las formas de onda (que representan una secuencia de números de 4 bits) de la Figura 13.8(b). La entrada D_0 es el bit menos significativo (LSB).

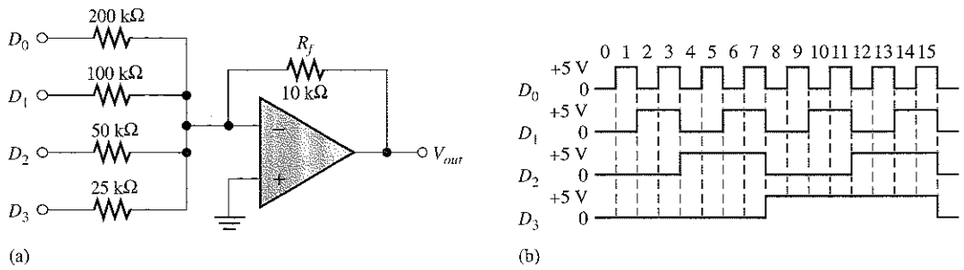


FIGURA 13.8

Solución. En primer lugar, se determina la corriente por cada una de las entradas ponderadas. Como la entrada inversora (–) del AO está a 0 V (tierra virtual) y un 1 binario corresponde a +5 V, la corriente que circula por cualquier resistencia de entrada es 5 V dividido por el valor de la resistencia:

$$I_0 = \frac{5\text{ V}}{200\text{ k}\Omega} = 0{,}025\text{ mA}$$

$$I_1 = \frac{5\text{ V}}{100\text{ k}\Omega} = 0{,}05\text{ mA}$$

$$I_2 = \frac{5\text{ V}}{50\text{ k}\Omega} = 0{,}1\text{ mA}$$

$$I_3 = \frac{5\text{ V}}{25\text{ k}\Omega} = 0{,}2\text{ mA}$$

Por la entrada inversora del AO casi no circula corriente, debido a su muy alta impedancia. Por tanto, suponemos que toda la corriente atraviesa la resistencia de realimentación R_f. Como un extremo de R_f está a 0 V (tierra virtual), la caída en R_f es igual a la tensión de salida, que es negativa con respecto a la tierra virtual.

$$V_{out(D0)} = (10\text{ k}\Omega)(-0{,}025\text{ mA}) = -0{,}25\text{ V}$$
$$V_{out(D1)} = (10\text{ k}\Omega)(-0{,}05\text{ mA}) = -0{,}5\text{ V}$$
$$V_{out(D2)} = (10\text{ k}\Omega)(-0{,}1\text{ mA}) = -1\text{ V}$$
$$V_{out(D3)} = (10\text{ k}\Omega)(-0{,}2\text{ mA}) = -2\text{ V}$$

En la Figura 13.8(b), el primer código de entrada binario es 0000, que produce una tensión de salida de 0 V. El siguiente código de entrada es 0001, que da lugar a una tensión de salida de –0,25 V. El siguiente código es 0010, que produce una tensión de salida de –0,5 V. El siguiente código es 0011, que da lugar a una tensión de salida de –0,25V + –0,5 V = –0,75V. Cada sucesivo código binario aumenta la tensión de salida en –0,25 V, por lo que, para esta secuencia binaria particular en las entradas, la salida es una forma de onda en escalera que va desde 0 V a –3,75 V, a escalones de –0,25 V. Esto se muestra en la Figura 13.9.

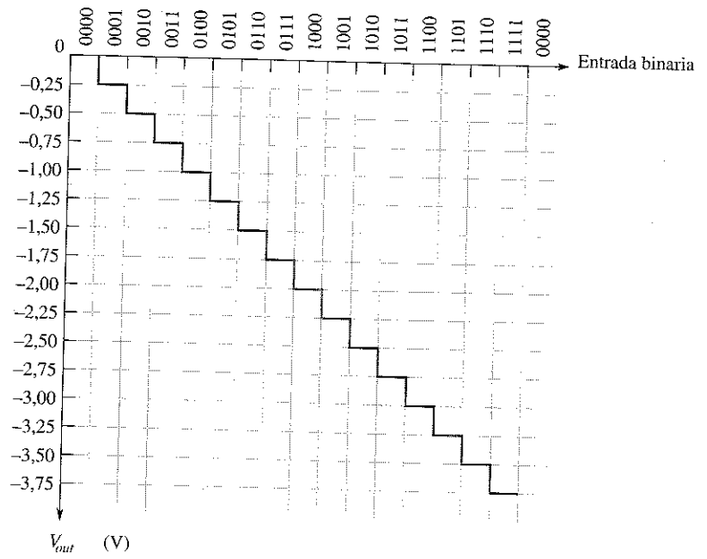


FIGURA 13.9
Salida del DAC de la Figura 13.8.

Problema relacionado. Invertir las formas de onda de entrada del DAC de la Figura 13.8 (D_3 a D_0, D_2 a D_1, D_1 a D_2, D_0 a D_3) y determinar la salida.

Convertidor digital-analógico en escalera *R/2R*

Otro método para realizar la conversión D/A es utilizar la red escalonada *R/2R*, como muestra la Figura 13.10 para el caso de cuatro bits. Este método resuelve uno de los problemas del DAC con ponderación binaria, ya que sólo requiere dos valores de resistencia.

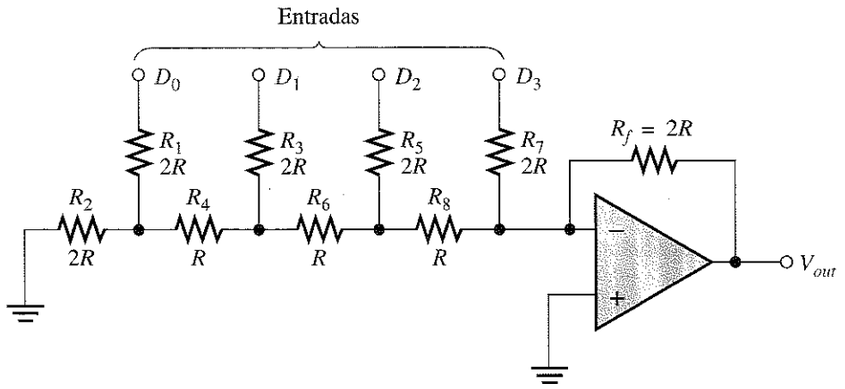


FIGURA 13.10
DAC en escalera R/2R.

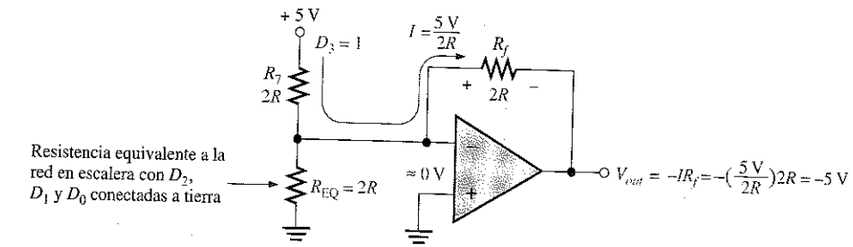
Comencemos asumiendo que la entrada D_3 está a nivel ALTO (+5 V) y las demás a nivel BAJO (tierra, 0 V). Esta condición representa el número binario 1000. Un análisis del circuito demostraría que este circuito es equivalente al mostrado en la Figura 13.11(a). A través de la resistencia equivalente $2R$ prácticamente no circula corriente, ya que la entrada inversora está a tierra virtual. Luego toda la corriente (I = 5 V/2R) que circula a través de R_7 pasa también por R_f, y la tensión de salida es –5 V. El amplificador operacional mantiene la entrada inversora (–) a casi cero voltios (≈ 0 V) debido a la realimentación negativa. Por tanto, toda la corriente pasa a través de R_f en lugar de por la entrada inversora.

La Figura 13.11(b) muestra el circuito equivalente cuando la entrada D_2 está a +5 V y las demás están a tierra. Esta condición representa 0100. Si se aplica el equivalente de Thevenin mirando desde R_8, se obtienen 2,5 V en serie con R, como se indica. Esto da lugar a una corriente a través de R_f de I = 2,5 V/2R, lo que determina una tensión de salida de –2,5 V. No olvidemos que no circula corriente por la entrada inversora ni por la resistencia equivalente conectada a tierra, ya que caen 0 V en ella debido a la tierra virtual.

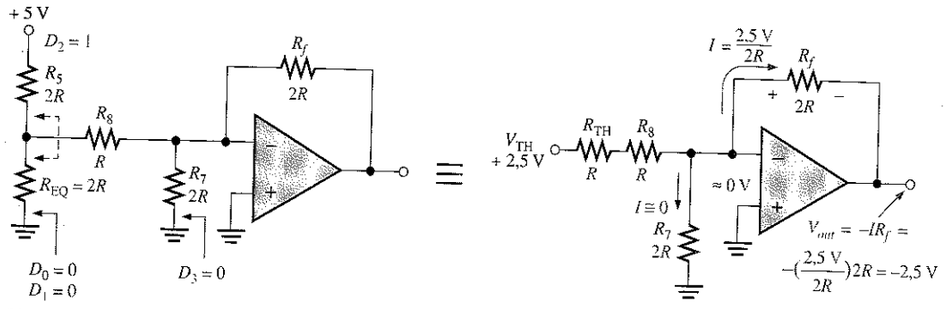
La Figura 13.11(c) muestra el circuito equivalente cuando la entrada D_1 está a + 5 V y las demás están a tierra. Esta condición representa el código 0010. De nuevo, aplicando el teorema de Thevenin al circuito a la izquierda de R_8, se obtienen 1,25 V en serie con R, como se indica. Esto da lugar a una corriente a través de R_f de I = 1,25 V/2R, lo que genera una tensión de salida de –1,25 V.

En la parte (d) de la Figura 3.11 se representa el circuito equivalente para el caso en que D_0 está a + 5 V y las demás entradas están a tierra. Esta condición representa el código 0001. Aplicando el teorema de Thevenin desde R_8, se obtienen 0,625 V en serie con R, como se indica. Esto da lugar a una corriente a través de R_f de I = 0,625 V/2R, lo que genera una tensión de salida de –0,625 V.

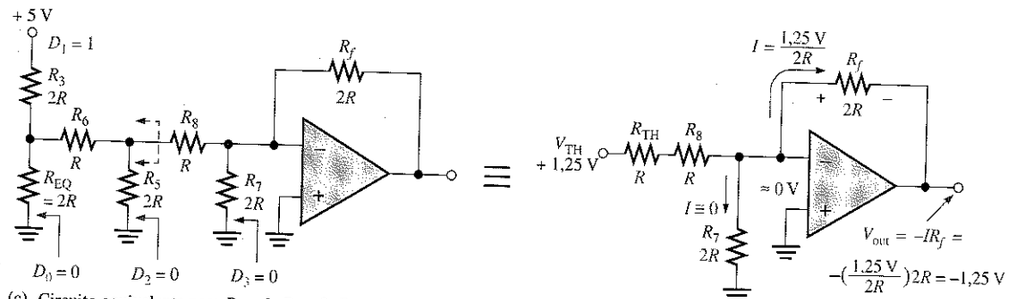
Obsérvese que cada entrada de menor peso sucesiva produce una tensión de salida que es la mitad de la anterior, por lo que la tensión de salida es proporcional al peso binario de los bits de entrada.



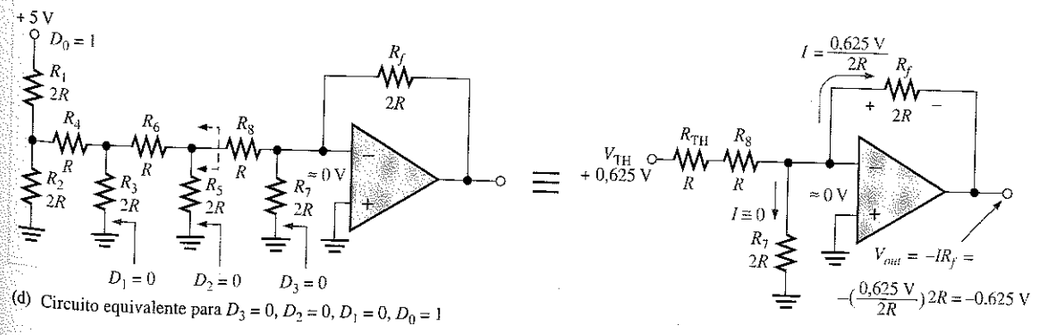
(a) Circuito equivalente para $D_3 = 1, D_2 = 0, D_1 = 0, D_0 = 0$



(b) Circuito equivalente para $D_3 = 0, D_2 = 1, D_1 = 0, D_0 = 0$



(c) Circuito equivalente para $D_3 = 0, D_2 = 0, D_1 = 1, D_0 = 0$



(d) Circuito equivalente para $D_3 = 0, D_2 = 0, D_1 = 0, D_0 = 1$

FIGURA 13.11
Análisis del DAC en escalera R/2R.

Características de funcionamiento de los convertidores analógico-digitales

Las prestaciones básicas de un DAC incluyen la resolución, precisión, linealidad, monotonicidad y tiempo de establecimiento, cada una de las cuales vamos a ver a continuación.

- *Resolución.* La resolución de un DAC es el recíproco del número de escalones discretos en la salida. Por supuesto, depende del número de bits de entrada. Por ejemplo, un DAC de 4 bits tiene una resolución de 1 entre $2^4 - 1$ (uno entre quince). Si se expresa como un porcentaje, será (1/15)100 = 6,67%. El número total de escalones discretos es igual a $2^n - 1$, donde n es el número de bits. La resolución también puede expresarse como el número de bits que se convierten.
- *Precisión.* La precisión es una comparación entre la salida real de un DAC y la salida esperada. Se expresa como un porcentaje de la tensión de salida a fondo de escala o máxima. Por ejemplo, si la salida a fondo de escala de un convertidor es 10 V y la precisión es ±0,1%, entonces el error máximo para cualquier tensión de salida es (10 V) (0,001) = 10 mV. Idealmente, la precisión debería ser, como mucho, ±1/2 del bit menos significativo. Para un convertidor de 8 bits, el bit menos significativo es el 0,39% del valor a fondo de escala. La precisión debería ser aproximadamente del ±0,2%.
- *Linealidad.* Un error lineal es una desviación de la salida ideal (una línea recta) del DAC. Un caso especial es el error de offset, que es la tensión de salida cuando los bits de entrada son todos cero.
- *Monotonicidad.* Un DAC es **monotónico** si no produce escalones invertidos cuando se le aplica secuencialmente su rango completo de bits de entrada.
- *Tiempo de asentamiento.* Normalmente, el tiempo de asentamiento se define como el tiempo que tarda un DAC en quedar dentro de ±1/2 LSB del valor final, cuando se produce un cambio en el código de entrada.

EJEMPLO 13.2 Determinar la resolución de los siguientes dispositivos DAC, expresada en porcentaje

(a) DAC de 8 bits **(b)** DAC de 12 bits.

Solución

(a) Para el convertidor de 8 bits,

$$\frac{1}{2^8 - 1} \times 100 = \frac{1}{255} \times 100 = \mathbf{0{,}392\%}$$

(b) Para el convertidor de 12 bits,

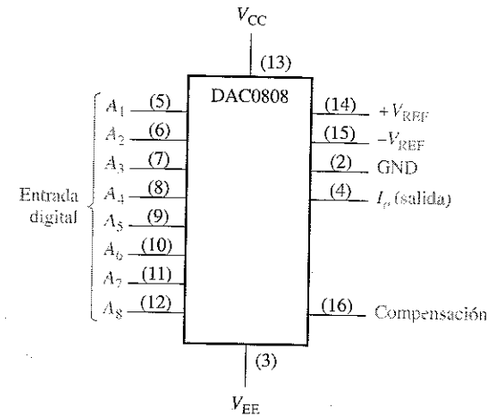
$$\frac{1}{2^{12} - 1} \times 100 = \frac{1}{4095} \times 100 = \mathbf{0{,}0244\%}$$

Problema relacionado. Calcular la resolución de un DAC de 16 bits.

Un convertidor digital-análogico específico

El DAC0808 es un ejemplo de un DAC en escalera R/2R. En la Figura 13.12 se muestra su diagrama de bloques. Este dispositivo opera con una tensión de alimentación continua V_{CC} en el rango comprendido entre +4,5 V y +5,5 V y una tensión V_{EE} en el rango de –4,5 V a -16,5 V. La resolución es de 8 bits, con un tiempo de asentamiento de salida de 150 ns. El tiempo de asentamiento es el tiempo que la salida analógica tarda en estabilizarse después de haber aplicado la entrada digital. El DAC0808 puede conectarse directamente a dispositivos TTL o CMOS y su precisión es del +0,19%.

FIGURA 13.12
Convertidor digital-análogico DAC0808.



En una aplicación típica, el DAC0808 se conecta a un amplificador operacional, como se muestra en la Figura 13.13, y la salida analógica se toma en dicho amplificador operacional. La entrada $+V_{REF}$ establece la tensión de salida analógica. Por ejemplo, una tensión de referencia de 10 V da lugar a una tensión de salida máxima de 10 V, como se muestra en la figura.

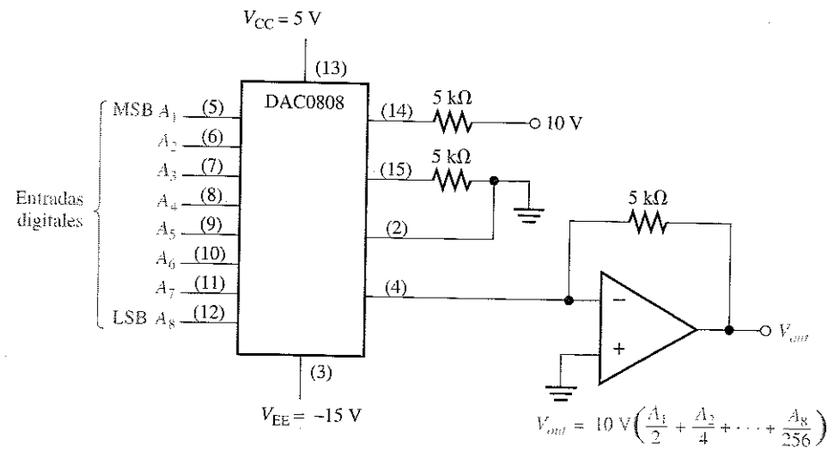


FIGURA 13.13
El DAC0808 en una configuración típica de convertidor DAC.

REVISIÓN DE LA SECCIÓN 13.2

1. ¿Cuál es la desventaja del DAC con ponderación binaria?
2. ¿Cuál es la resolución de un DAC de 4 bits?

13.3 ■ CONVERSIÓN ANALÓGICA-DIGITAL (A/D)

Como hemos visto, la conversión analógica-digital es el proceso por el cual una magnitud analógica se convierte a formato digital. La conversión A/D es necesaria cuando se debe expresar en forma digital una serie de magnitudes medidas, para procesarlas en una computadora, presentarlas en un display o almacenarlas. A continuación, vamos a examinar varios tipos de convertidores analógico-digitales (ADC).

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar cómo funciona un ADC flash. ■ Describir el funcionamiento de un ADC de rampa en escalera. ■ Describir el funcionamiento de un ADC de seguimiento. ■ Explicar los ADC de pendiente simple y de pendiente doble. ■ Describir el funcionamiento de un ADC por aproximaciones sucesivas.

Convertidor analógico-digital flash (paralelo)

El método flash utiliza comparadores que comparan una serie de tensiones de referencia con la tensión de entrada analógica. Cuando la tensión analógica sobrepasa a la tensión de referencia de un comparador determinado, se genera un nivel ALTO. La Figura 13.14 presenta un convertidor de 3 bits que utiliza siete circuitos comparadores; no se necesita comparador para el caso de que todas las comparaciones sean cero. En general, se requieren 2^n–1 comparadores para la conversión a un código binario de n bits. Una de las desventajas del **ADC flash** es el gran número de comparadores necesarios para un número binario de tamaño razonable. Su principal ventaja es que tiene un tiempo de conversión rápido.

La tensión de referencia de cada comparador se establece mediante un circuito divisor de tensión. La salida de cada comparador se conecta a una entrada del codificador de prioridad. El codificador se muestrea mediante un impulso aplicado a la entrada de habilitación, y el código de tres bits que representa el valor de la entrada analógica se presenta en las salidas del codificador. El código binario queda determinado por la entrada de mayor orden que se encuentre a nivel ALTO.

La velocidad de muestreo determina la precisión con la que la secuencia de códigos digitales representa la entrada analógica del ADC. Cuanto más muestras se toman en una determinada unidad de tiempo, más precisa es la señal digital que representa a la señal analógica.

El Ejemplo 13.3 ilustra el funcionamiento básico del ADC flash de la Figura 13.14.

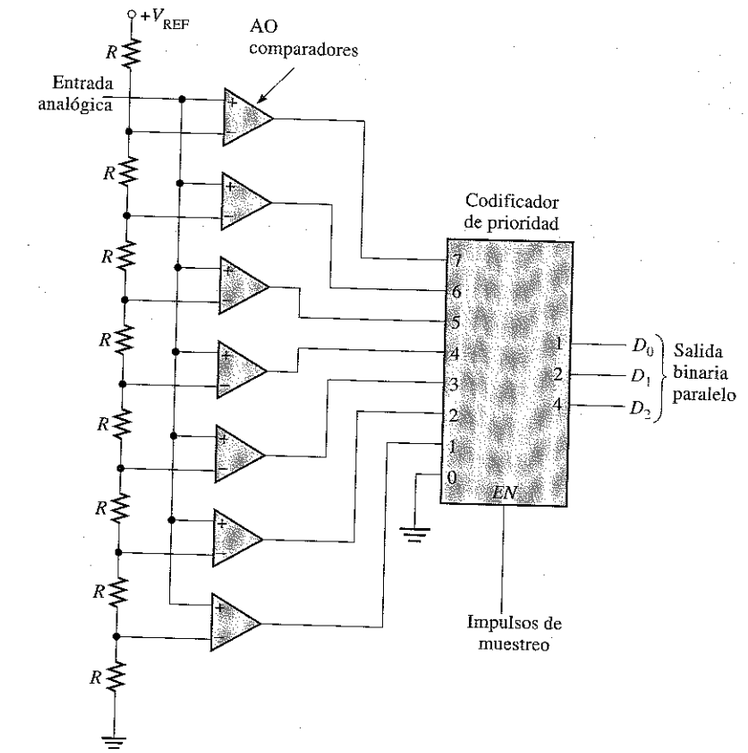


FIGURA 13.14
ADC flash de 3 bits.

EJEMPLO 13.3 Determinar el código binario de salida del ADC flash de tres bits para la señal analógica de entrada de la Figura 13.15 y los impulsos de muestreo indicados (habilitación del codificador). En este ejemplo, $V_{REF} = +8$ V.

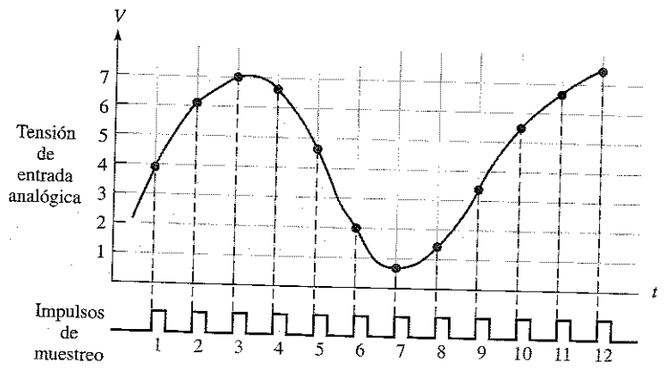


FIGURA 13.15
Muestreo de una señal analógica para convertirla a formato digital.

Solución. La secuencia digital de salida resultante corresponde a los valores siguientes y al cronograma que se muestra en la Figura 13.16 en relación con los impulsos de muestreo.

100, 110, 111, 110, 100, 010, 000, 001, 011, 101, 110, 111

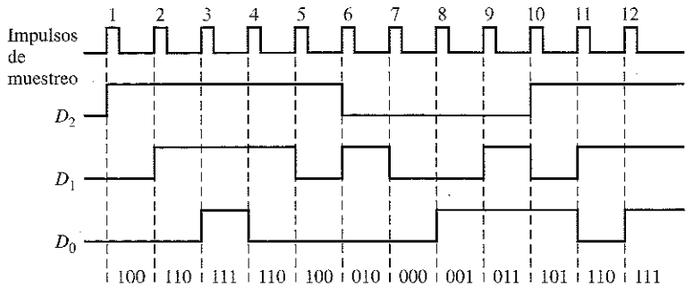


FIGURA 13.16
Salidas digitales resultantes para los valores muestreados. La salida D_0 es el LSB.

Problema relacionado. Si se duplica la frecuencia de los impulsos de muestreo de la Figura 13.15, determinar los números binarios representados por la secuencia digital de salida resultante, para 24 impulsos.

Convertidor analógico-digital de rampa en escalera

El método de rampa en escalera para la conversión A/D se conoce también como método *de rampa digital* o método *contador*. Se emplea un DAC y un contador binario para generar el valor digital correspondiente a una entrada analógica. La Figura 13.17 presenta un diagrama de este tipo de convertidor.

Suponemos que el contador se inicializa en el estado RESET y la salida del DAC es cero. A continuación, suponemos que se aplica una tensión analógica a la entrada. Cuando se sobrepasa la tensión de referencia (salida del DAC), el comparador conmuta su salida a nivel ALTO y activa la puerta AND. Los impulsos de reloj hacen que el contador avance a través de sus estados binarios, produciendo una tensión de referencia en escalera a partir del DAC. El contador continúa pasando de un estado al siguiente, dando lugar sucesivamente a escalones más altos en la tensión de referencia. Cuando la tensión de referencia en escalera alcanza la tensión de entrada analógica, la salida del comparador pasa a nivel BAJO y la puerta AND se desactiva, lo que hace que cesen los impulsos de reloj y el contador se pare. El estado binario del contador en este momento es igual al número de escalones de la tensión de referencia necesarios para hacer que dicha tensión de referencia sea igual o mayor que la entrada analógica. Por supuesto, este número binario representa el valor de la entrada analógica. La lógica de control carga la cuenta binaria en los latches y pone a cero el contador, iniciándose después otra secuencia de cuenta para muestrear el valor de entrada.

NOTAS INFORMÁTICAS
Las tarjetas de sonido que se utilizan en las computadoras emplean un ADC para convertir el sonido procedente de un micrófono, un reproductor de CD de audio u otra fuente en sonido digital. El ADC envía la señal digital a un procesador digital de señal (DSP). Basándose en las instrucciones almacenadas en una ROM, una función del DSP consiste en comprimir la señal digital con el fin de usar menos espacio de almacenamiento. El DSP envía entonces los datos comprimidos al procesador de la computadora que, a su vez, envía los datos a un disco duro o CD-ROM para su almacenamiento. Para reproducir un sonido grabado, el procesador recupera los datos almacenados y los envía al DSP, donde se descomprimen, siendo enviados después a un DAC. La salida del DAC, que es una reproducción de la señal sonora original, se aplica a los altavoces.

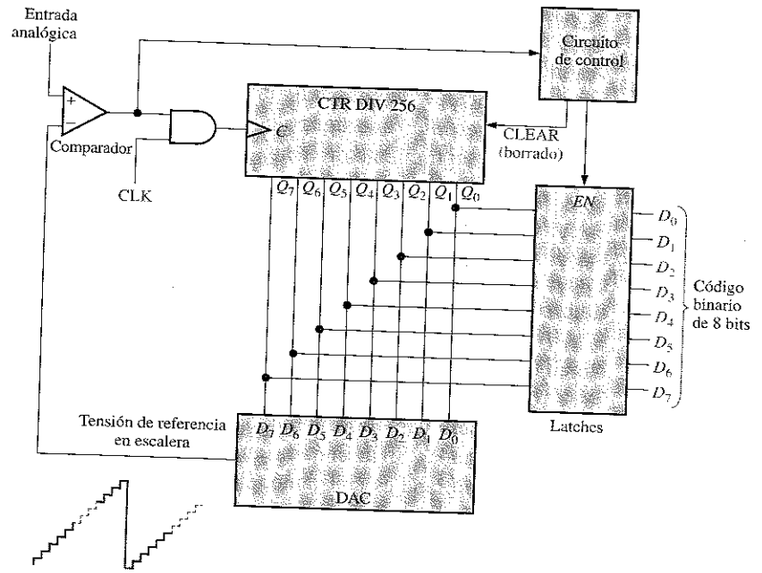


FIGURA 13.17
ADC de rampa digital (8 bits).

El método de rampa en escalera es más lento que el método flash porque, en el caso peor de entrada máxima, el contador debe pasar a través del número máximo de estados antes de realizar la conversión. Para una conversión de 8 bits, esto significa un máximo de 256 estados. La Figura 13.18 ilustra una secuencia de conversión con 4 bits. Obsérvese que, para cada muestra, el contador debe contar desde cero hasta el escalón en el que la tensión de referencia alcanza a la tensión de entrada analógica. El tiempo de conversión es variable, en función de la tensión analógica.

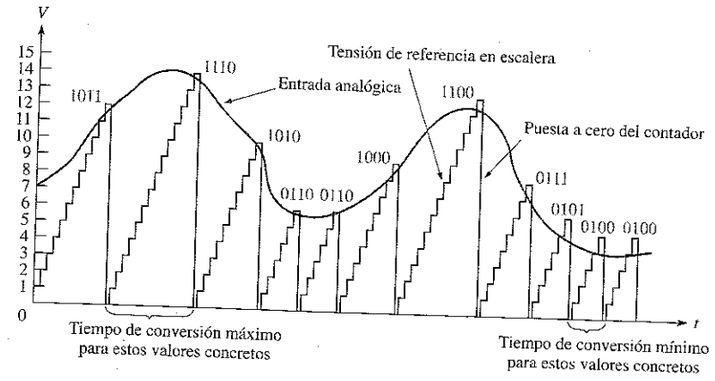


FIGURA 13.18
Ejemplo de una conversión de 4 bits, que muestra una señal de entrada analógica y la tensión de referencia en escalera para un ADC de rampa digital.

Convertidor analógico-digital de seguimiento

El método de seguimiento utiliza un contador ascendente/descendente y es más rápido que el método de rampa digital, porque el contador no se pone a cero después de cada muestreo, sino que sigue a la entrada analógica. La Figura 13.19 muestra un típico ADC de seguimiento de 8 bits.

Mientras que la tensión de referencia de salida es menor que la entrada analógica, la salida del comparador permanece a nivel ALTO, poniendo el contador en modo ascendente (UP), lo que da lugar a una secuencia ascendente de cuenta binaria. Esto hace que la tensión de salida en escalera del DAC se incremente hasta que la rampa alcance el valor de la tensión de entrada.

Cuando la tensión de referencia es igual a la entrada analógica, la salida del comparador conmuta a nivel BAJO y pone al contador en modo descendente (DOWN), haciendo que la cuenta se decremente. Si la entrada analógica decrece, el contador continuará descendiendo a través de su secuencia y seguirá de forma efectiva a la entrada. Si la entrada aumenta, el contador volverá al modo ascendente. Cuando la entrada es constante, el contador desciende una vez cuando se alcanza la tensión de entrada. Ahora, la salida de referencia es menor que la entrada analógica, y la salida del comparador se pone a nivel ALTO, haciendo que el contador pase a modo ascendente. Tan pronto como el contador asciende al estado correspondiente, la tensión de referencia se hace mayor que la entrada, conmutando el comparador a nivel BAJO. Esto hace que el contador se decremente una unidad. Esta acción de avance-retroceso continúa mientras la entrada analógica sea un valor constante, lo que origina una oscilación de salida entre dos estados binarios. Ésta es la desventaja de este tipo de convertidor.

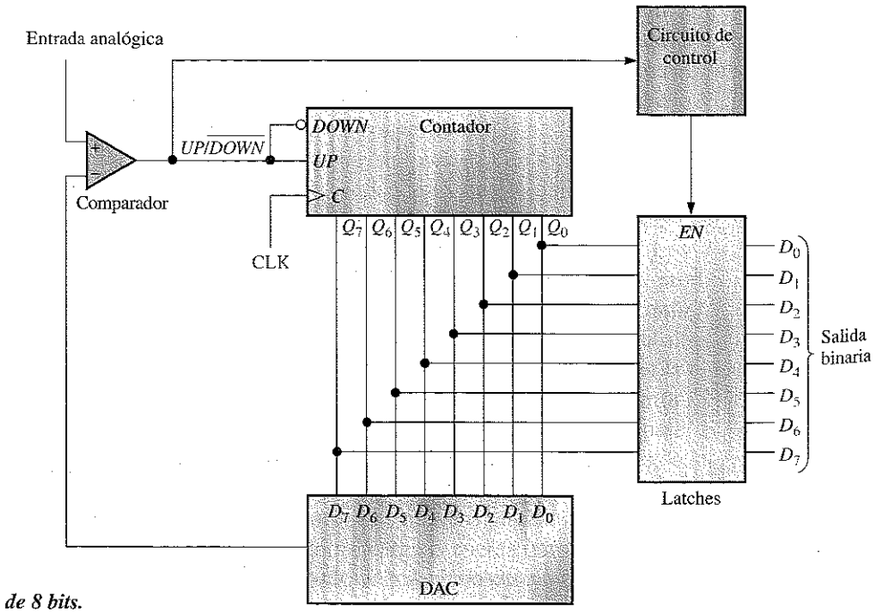


FIGURA 13.19
ADC de seguimiento de 8 bits.

La Figura 13.20 ilustra la acción de seguimiento de este tipo de ADC para una conversión de 4 bits.

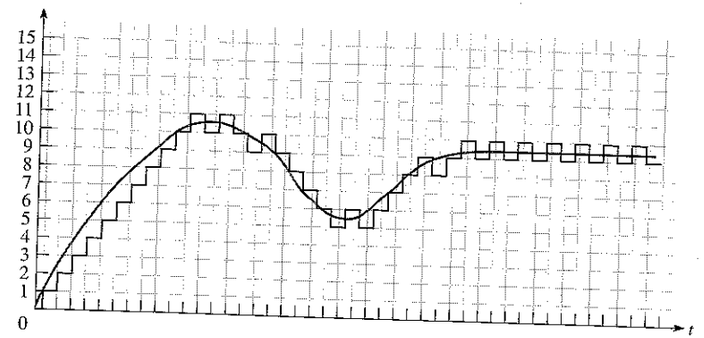


FIGURA 13.20
Acción de seguimiento de un ADC de seguimiento.

Convertidor analógico-digital de pendiente simple

A diferencia de los métodos de rampa en escalera y seguimiento, el convertidor de pendiente simple no requiere un DAC. Se utiliza un generador de rampa lineal para generar una tensión de referencia de pendiente constante. En la Figura 13.21 se muestra un diagrama. Al comienzo del ciclo de conversión, el contador está en estado RESET y la salida del generador de rampa es 0 V. En esta situación, la entrada analógica es mayor que la tensión de referencia y, por tanto, se produce un nivel alto en la salida del comparador. Este nivel alto habilita la señal de reloj para el contador y arranca el generador de rampa.

Supongamos que la pendiente de la rampa es 1 V/ms. La entrada aumentará hasta que sea igual a la entrada analógica; en dicho momento la rampa se pone a cero y la cuenta binaria o BCD se almacena en los latches mediante la lógica de control. Supongamos que la entrada analógica es 2 V cuando se produce la igualdad de ambas señales. Esto significa que la rampa también es 2 V y que han transcurrido 2 ms. Puesto que la salida del comparador durante 2 ms ha estado a nivel ALTO, se ha permitido el paso de 200 impulsos de reloj a través de la puerta del contador (suponemos que la frecuencia de reloj es 100 kHz). Cuando se realiza la comparación, el contador está en el estado binario que representa el decimal 200. Aplicando la decodificación y escalado adecuados, este número binario se puede presentar en un display como 2,00 V. Este concepto básico es el que utilizan algunos voltímetros digitales.

Convertidor analógico-digital de doble pendiente

El funcionamiento del ADC de doble pendiente es similar al de pendiente simple, excepto en que se utiliza una rampa de pendiente variable y otra de pendiente fija. Este tipo de convertidor se utiliza comúnmente en voltímetros digitales y otros tipos de instrumentos de medida.

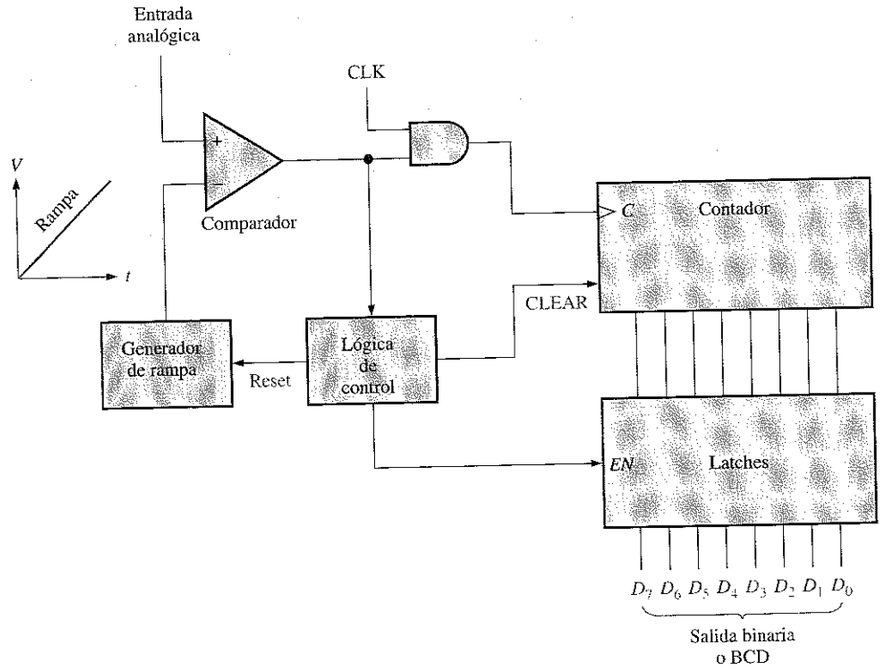


FIGURA 13.21
ADC de pendiente simple.

Se utiliza un generador de rampa (integrador), A_1, para generar la característica de pendiente doble. En la Figura 13.22 se presenta un esquema de un ADC de pendiente doble como referencia.

La Figura 13.23 ilustra la conversión de pendiente doble. Comencemos suponiendo que el contador está en estado RESET y la salida del integrador es cero. Ahora supongamos que se aplica a la entrada una tensión de entrada positiva por medio del interruptor (SW), comandado por la lógica de control. Puesto que la entrada inversora de A_1 está a tierra virtual y suponiendo que V_{in} es constante durante un periodo de tiempo, circulará una corriente constante a través de la resistencia de entrada R y, por tanto, a través del condensador C. El condensador C se cargará linealmente ya que la corriente es constante y, como resultado, habrá una rampa lineal negativa de tensión en la salida de A_1, como se muestra en la Figura 13.23(a).

Cuando el contador alcance un valor de cuenta especificado, se pondrá a cero (RESET) y la lógica de control conmutará a la tensión de referencia negativa ($-V_{REF}$) aplicándola a la entrada de A_1, como muestra la Figura 13.23(b). En este instante, el condensador está cargado a una tensión negativa ($-V$) proporcional a la tensión analógica de entrada.

A continuación, el condensador se descarga linealmente debido a la corriente constante procedente de $-V_{REF}$, como ilustra la Figura 13.23(c). Esta descarga lineal produce una rampa positiva en la salida de A_1, cuyo valor inicial es $-V$ y que tiene una

pendiente constante, independiente de la tensión de carga. A medida que el condensador se descarga, el contador avanza desde su estado de RESET. El tiempo que tarda el condensador en descargarse hasta 0 V depende de la tensión inicial $-V$ (proporcional a V_{in}), puesto que la velocidad (pendiente) de descarga es constante. Cuando la tensión de salida del integrador (A_1) alcanza el valor cero, el comparador A_2 conmuta al estado BAJO e inhabilita la señal de reloj aplicada al contador. La cuenta binaria se almacena en los latches, completando un ciclo de conversión. La cuenta binaria es proporcional a V_{in}, ya que el tiempo que tarda el condensador en descargarse sólo depende de $-V$, y el contador registra este intervalo de tiempo.

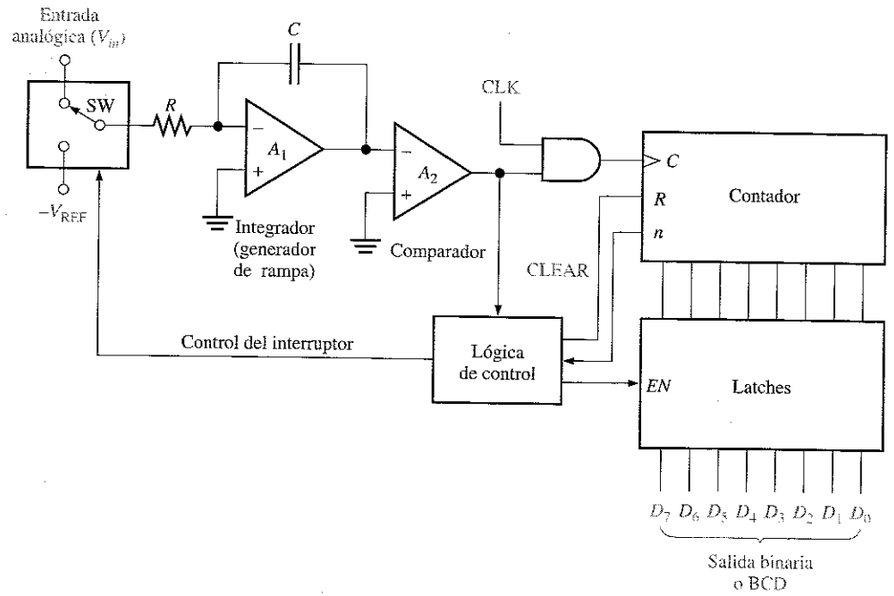


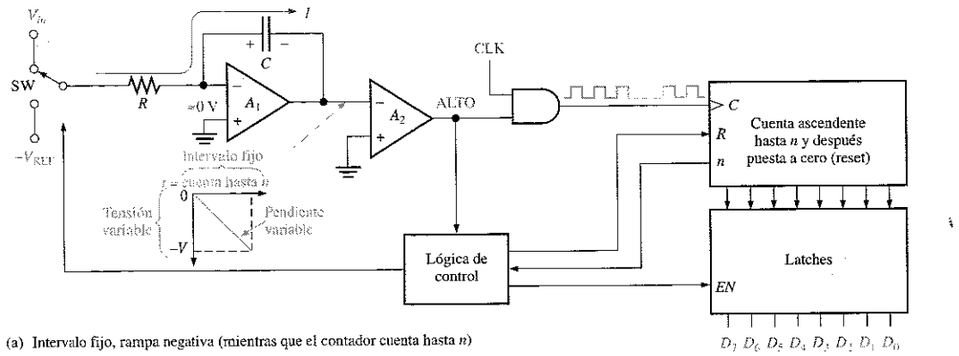
FIGURA 13.22
ADC de pendiente doble.

Convertidor analógico-digital por aproximaciones sucesivas

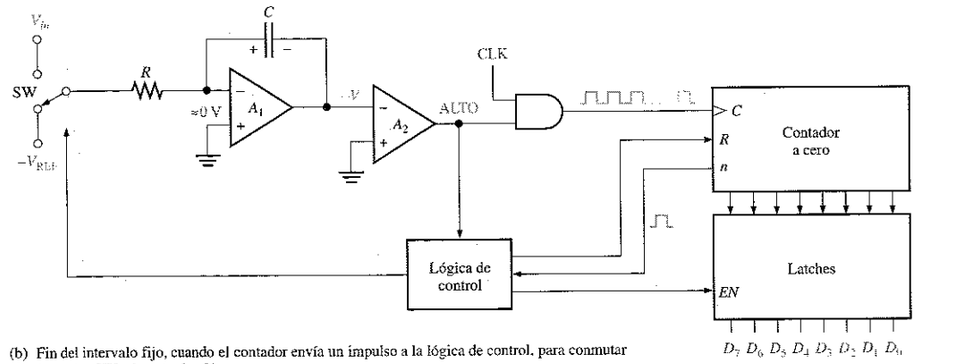
Quizás el método de conversión A/D más ampliamente utilizado es el de las aproximaciones sucesivas. Tiene un tiempo de conversión mucho menor que los otros métodos, a excepción del método flash. Asimismo, el tiempo de conversión es fijo para cualquier valor de la entrada analógica.

La Figura 13.24 muestra un diagrama de bloques básico de un ADC por aproximaciones sucesivas de 4 bits. Está formado por un DAC, un registro de aproximaciones sucesivas (SAR, successive-approximation register) y un comparador. Su funcionamiento básico es el siguiente: los bits de entrada al DAC se habilitan (se ponen a 1) de uno en uno sucesivamente, comenzando por el bit más significativo (MSB). Cada vez que se habilita un bit, el comparador produce una salida que indica si la tensión analógica de entrada es mayor o menor que la salida del DAC. Si la salida del DAC es

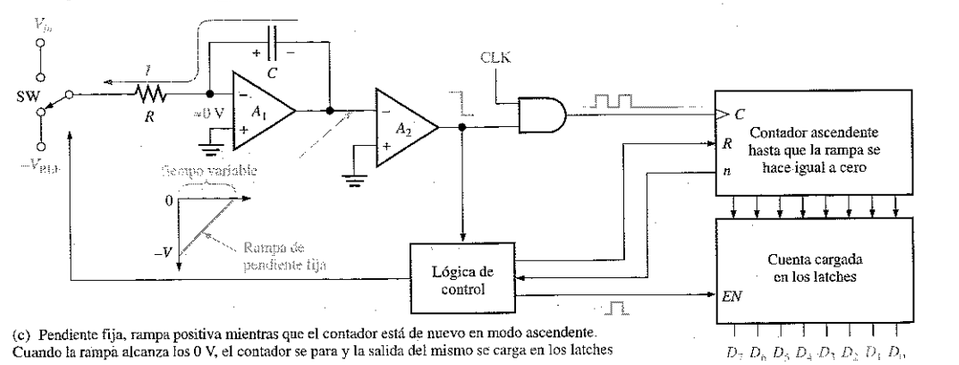
mayor que la entrada analógica, la salida del comparador está a nivel BAJO, haciendo que el bit en el registro pase a cero. Si la salida es menor que la entrada analógica, el bit 1 se mantiene en el registro. El sistema realiza esta operación con el MSB primero, luego con el siguiente bit más significativo, después con el siguiente, y así sucesivamente. Después de que todos los bits del DAC hayan sido aplicados, el ciclo de conversión estará completo.



(a) Intervalo fijo, rampa negativa (mientras que el contador cuenta hasta *n*)



(b) Fin del intervalo fijo, cuando el contador envía un impulso a la lógica de control, para conmutar el interruptor SW a la entrada $-V_{REF}$



(c) Pendiente fija, rampa positiva mientras que el contador está de nuevo en modo ascendente. Cuando la rampa alcanza los 0 V, el contador se para y la salida del mismo se carga en los latches

FIGURA 13.23
Conversión de doble pendiente.

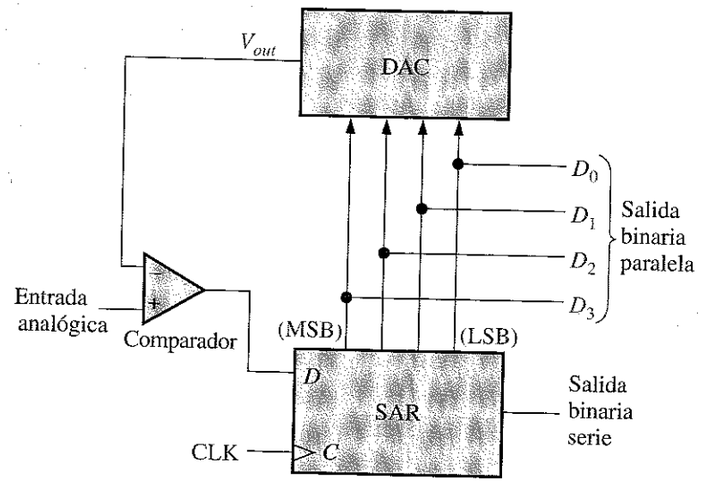


FIGURA 13.24
ADC por aproximaciones sucesivas.

Para comprender mejor el funcionamiento del ADC por aproximaciones sucesivas, vamos a realizar un ejemplo específico de una conversión de 4 bits. La Figura 13.25 ilustra la conversión paso a paso de una tensión analógica de entrada constante (en este caso, 5,1 V). Suponemos que el DAC tiene las siguientes características de salida: $V_{out} = 8$ V para el bit 2^3 (MSB), $V_{out} = 4$ V para el bit 2^2, $V_{out} = 2$ V para el bit 2^1 y $V_{out} = 1$ V para el bit 2^0 (LSB).

La Figura 13.25(a) muestra el primer paso del ciclo de conversión con el MSB = 1. La salida del DAC es 8 V. Puesto que es mayor que la entrada analógica de 5,1 V, la salida del comparador está a nivel BAJO, lo que hace que el MSB del SAR se ponga a 0.

La Figura 13.25(b) muestra el segundo paso del ciclo de conversión, con el bit 2^2 igual a 1. La salida del DAC es 4 V. Puesto que es menor que la entrada analógica de 5,1 V, la salida del comparador conmuta a nivel ALTO, lo que hace que este bit se mantenga en el SAR.

La Figura 13.25(c) muestra el tercer paso del ciclo de conversión, con el bit 2^1 igual a 1. La salida del DAC es 6 V, ya que los bits de entrada 2^2 y 2^1 están a 1; 4 V + 2 V = 6 V. Puesto que es mayor que la entrada analógica de 5,1 V, la salida del comparador conmuta a nivel BAJO, lo que hace que este bit se ponga a cero.

La Figura 13.25(d) muestra el cuarto y último paso del ciclo de conversión, con el bit 2^0 igual a 1. La salida del DAC es 5 V, ya que los bits de entrada 2^2 y 2^0 están a 1; 4 V + 1 V = 5 V.

Se han probado los cuatro bits, y el ciclo de conversión ha sido completado. En este momento, el código binario almacenado en el registro es 0101, que es aproximadamente el valor binario de la entrada analógica de 5,1 V. Más bits producirán un resultado aún más preciso. A continuación, se inicia un nuevo ciclo de conversión y el proceso se repite. El SAR se borra al comienzo de cada nuevo ciclo.

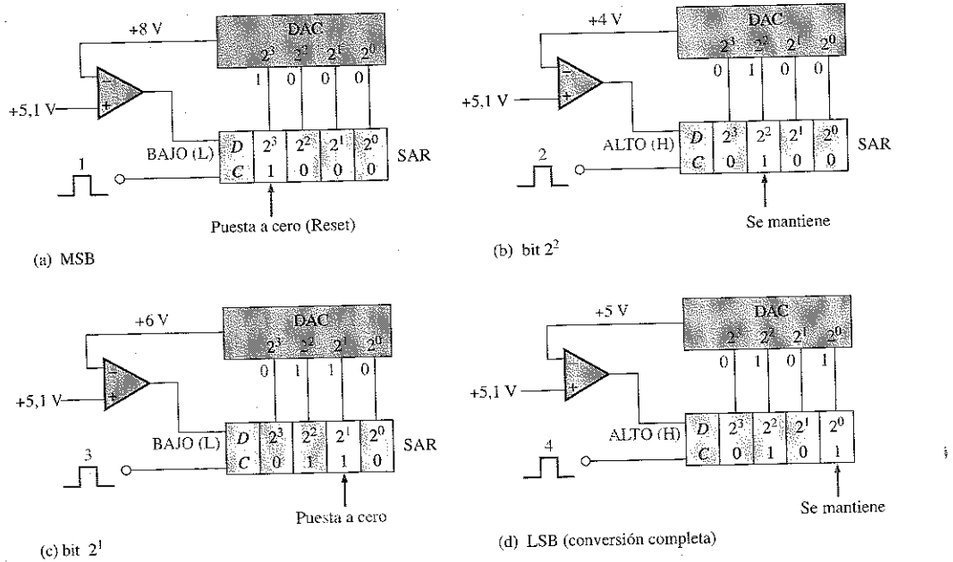


FIGURA 13.25
Proceso de conversión por aproximaciones sucesivas.

Un convertidor analógico-digital específico

El ADC0804 es un ejemplo de ADC por aproximaciones sucesivas. En la Figura 13.26 se presenta el diagrama de bloques. Este dispositivo funciona con una alimentación de +5 V y tiene una resolución de ocho bits, con un tiempo de conversión de 100 µs. También garantiza la característica de monotonicidad y dispone de un generador de reloj interno. Las salidas de datos triestado sirven para realizar la interfaz con el sistema de buses de un microprocesador.

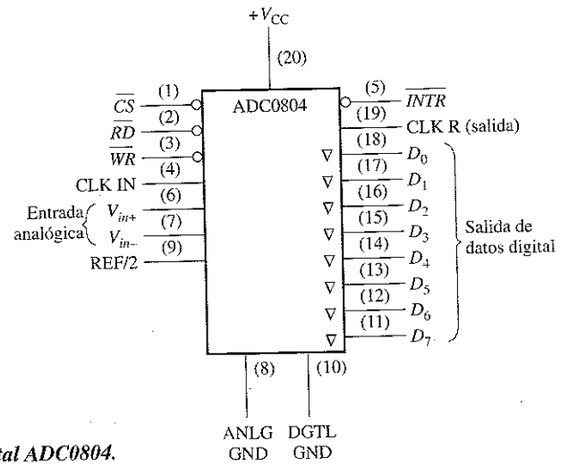


FIGURA 13.26
Convertidor analógico-digital ADC0804.

El funcionamiento básico del dispositivo es el siguiente. El ADC0804 contiene el equivalente a una red DAC de 256 resistencias. La lógica de aproximaciones sucesivas secuencia la red para adaptar la tensión analógica de entrada diferencial ($V_{in+} - V_{in-}$) a una salida de la red resistiva. En primer lugar, se comprueba el MSB. Después de realizar ocho comparaciones (sesenta y cuatro periodos de reloj), un código binario de 8 bits se transfiere a los latches de salida y la salida de interrupción ($\overline{INTR}$) pasa a nivel BAJO. El dispositivo puede funcionar en modo de conversión libre (free-running), conectando la salida $\overline{INTR}$ a la entrada de escritura ($\overline{WR}$) y manteniendo la entrada de inicio de conversión, $\overline{CS}$, a nivel BAJO. Para garantizar una adecuada inicialización bajo todas las posibles condiciones, se requiere un nivel BAJO en la entrada $\overline{WR}$ durante el ciclo de conexión de la alimentación. A partir de ahí, si se pone $\overline{CS}$ a nivel bajo en cualquier instante, se interrumpirá el proceso de conversión.

Cuando la entrada $\overline{WR}$ pasa a nivel BAJO, el registro de aproximaciones sucesivas (SAR) interno y el registro de desplazamiento de 8 bits se ponen a cero. Mientras $\overline{CS}$ y $\overline{WR}$ permanezcan a nivel BAJO, el ADC permanecerá en estado de RESET. El periodo de conversión se inicia de uno a ocho periodos de reloj después de que $\overline{CS}$ o $\overline{WR}$ hagan una transición de nivel BAJO a nivel ALTO.

Cuando ambas entradas $\overline{CS}$ y $\overline{RD}$ están a nivel BAJO, el latch de salida triestado se habilita y el código de salida se aplica a las líneas D_0 a D_7. Cuando la entrada $\overline{CS}$ o la entrada $\overline{RD}$ pasan a nivel ALTO, las salidas D_0 a D_7 se desactivan.

REVISIÓN DE LA SECCIÓN 13.3

1. ¿Cuál es el método más rápido de conversión analógica-digital?
2. ¿Qué método de conversión A/D utiliza un contador ascendente/descendente?
3. ¿Tiene el convertidor por aproximaciones sucesivas un tiempo de conversión fijo?

13.4 ■ LOCALIZACIÓN DE AVERÍAS EN LOS CONVERTIDORES DAC Y ADC

Las pruebas básicas de los convertidores DAC y ADC incluyen la comprobación de sus características de funcionamiento, tales como monotonicidad, offset, linealidad y ganancia, así como la comprobación de la existencia de códigos incorrectos o ausentes. En esta sección, se introducen los fundamentos para probar estas interfaces analógicas.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar cómo se prueban la monotonicidad, la no linealidad diferencial, la alta o baja ganancia y el error de offset en los DAC. ■ Explicar cómo se prueba si hay códigos ausentes o códigos incorrectos en los ADC y cómo se comprueba el valor de offset.

Comprobación de los convertidores digital-analógicos

En la Figura 13.27 se ilustra el concepto de comprobación de un DAC. En este método básico, se aplica a las entradas una secuencia de códigos binarios, y se observa la salida resultante. La secuencia de códigos binarios abarca todo el rango completo de valores de 0 a $2^n - 1$ en orden ascendente, siendo n el número de bits.

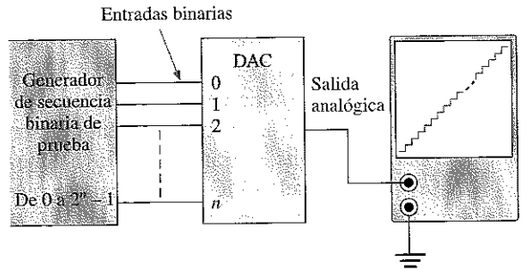


FIGURA 13.27
Configuración básica de prueba de un DAC.

La salida ideal es, como se indica, una escalera recta. Cuanto mayor es el número de bits del código binario, mejor es la resolución. Es decir, cuantos más escalones discretos hay, más se aproxima la salida a una rampa lineal.

Errores de la conversión D/A

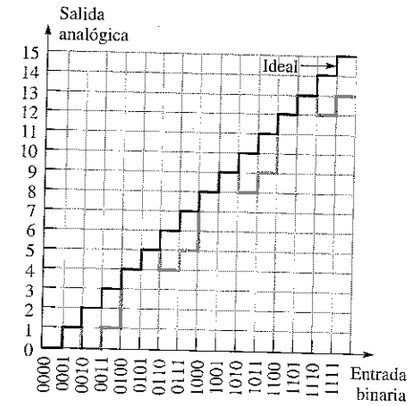
En la Figura 13.28 se muestran diversos errores potenciales de la conversión D/A. En la figura se utiliza una conversión de cuatro bits, con el fin de ilustrar estos errores. Una conversión de 4 bits produce quince escalones discretos. Cada una de las gráficas de la figura incluye la rampa en escalera ideal, para compararla con las salidas que presentan fallos.

No monotonicidad. Los escalones en sentido inverso de la Figura 13.28(a) indican un funcionamiento no monotónico, que es una forma de no linealidad. En este caso particular, el error se produce porque el bit 2^1 del código binario se interpreta como un 0 constante. Es decir, un cortocircuito hace que la línea de entrada del bit permanezca a nivel BAJO.

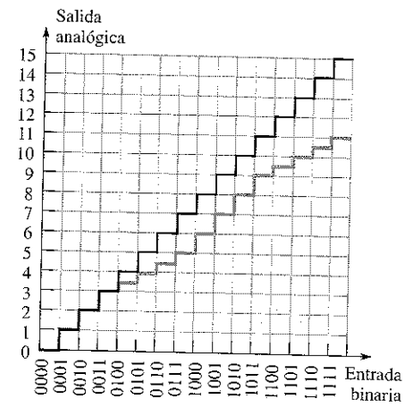
No linealidad diferencial. La Figura 13.28(b) ilustra la no linealidad diferencial, en la que la amplitud del escalón es menor de lo que debería ser para ciertos códigos de entrada. Esta salida en concreto podría ser producida por el peso insuficiente del bit 2^2, debido a un fallo de la resistencia de entrada. También podría haber escalones con amplitudes mayores que la normal, si un peso en particular fuera mayor de lo que debería ser.

Baja y alta ganancia. En la Figura 13.28(c) se ilustran los errores de alta y baja ganancia. En el caso de baja ganancia, todas las amplitudes de los escalones son menores que la ideal. En el caso de alta ganancia, todas las amplitudes de los escalones son mayores que la ideal. Esta situación puede deberse a un fallo de la resistencia de realimentación en el circuito del amplificador operacional.

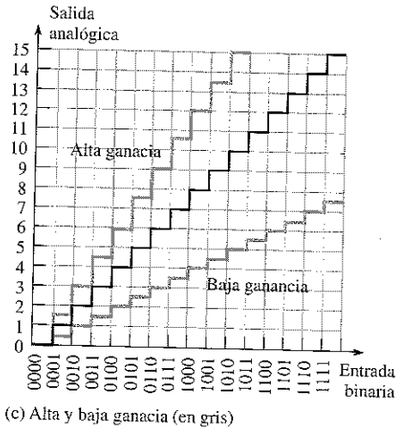
Error de offset. En la Figura 13.28(d) se ilustra el error de offset. Obsérvese que, cuando la entrada binaria es 0000, la tensión de salida no es cero, y este valor de offset es el mismo para todos los escalones de la conversión. En esta situación, un fallo del amplificador operacional puede ser el culpable.



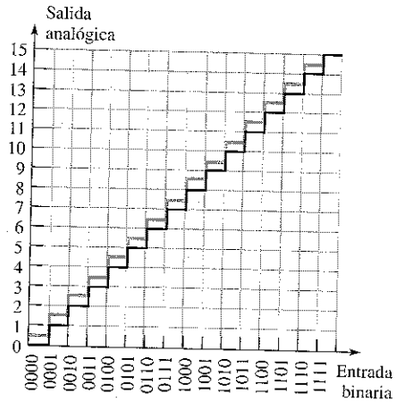
(a) Salida no monotónica (en gris)



(b) No linealidad diferencial (en gris)



(c) Alta y baja ganancia (en gris)



(d) Error de Offset (en gris)

FIGURA 13.28
Diversos errores de la conversión D/A.

EJEMPLO 13.4 Se observa la salida del DAC de la Figura 13.29 cuando se aplica a las entradas una secuencia binaria ascendente de 4 bits. Identificar el tipo de error y sugerir un método para aislar el fallo.

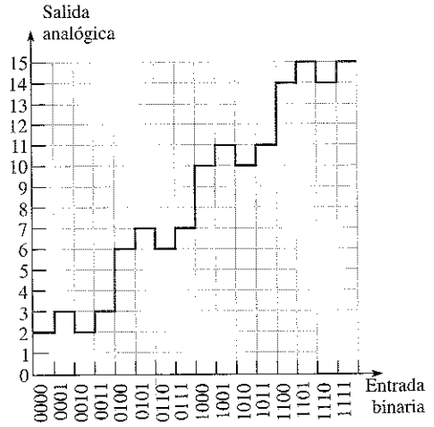
Solución. En este caso el DAC es no monotónico. El análisis de la salida revela que el dispositivo convierte la siguiente secuencia, en lugar de la secuencia binaria real aplicada a las entradas.

0010, 0011, 0010, 0011, 0110, 0111, 0110, 0111, 1010, 1011, 1010, 1011, 1110, 1111, 1110, 1111

Aparentemente, el bit 2^1 se mantiene en el estado ALTO (1). Para localizar el problema, en primer lugar se comprueba el pin de la entrada del bit del dispositivo. Si cam-

bia de estado, el fallo es interno, probablemente un circuito abierto. Si el pin externo no cambia de estado y está siempre a nivel ALTO, debe comprobarse si existe un cortocircuito externo a $+V$, que puede haber sido causado por un puente de soldadura en algún punto de la tarjeta.

FIGURA 13.29



Problema relacionado. Determinar la salida de un DAC cuando se aplica a la entrada una secuencia binaria de 4 bits lineal y el bit 2^0 permanece a nivel ALTO.

Comprobación de los convertidores analógico-digitales

En la Figura 13.30 se muestra un método para probar los convertidores ADC. Se utiliza un DAC como parte del equipo de pruebas, para convertir la salida del ADC de nuevo a formato analógico, con el fin de compararla con la señal de entrada de prueba.

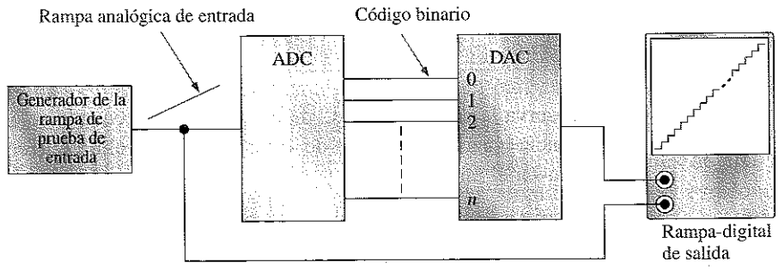


FIGURA 13.30
Método de comprobación de los convertidores ADC.

A la entrada del ADC se aplica una entrada de prueba en forma de rampa lineal. La secuencia binaria de salida resultante se aplica entonces al DAC y se convierte en una

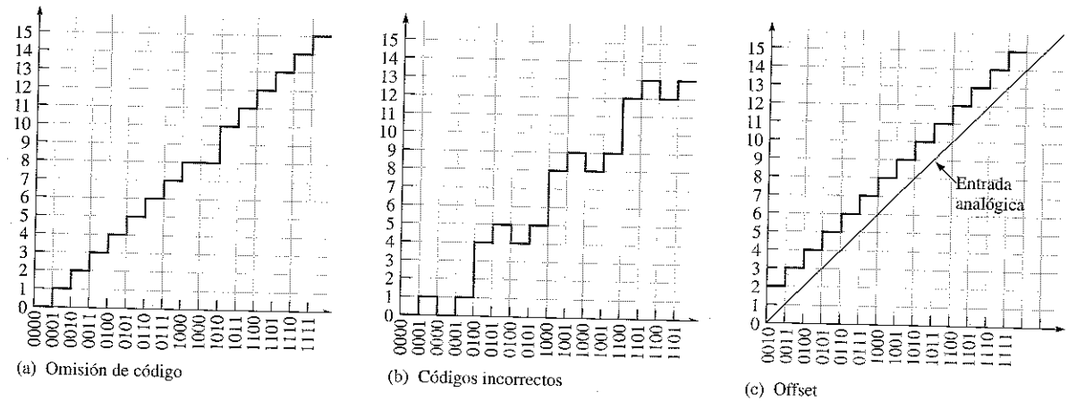
rampa escalonada. Las rampas de entrada y salida se comparan, en busca de cualquier desviación.

Errores de la conversión A/D

De nuevo utilizamos una conversión de 4 bits para ilustrar los principios. Supongamos que la entrada de prueba es una rampa lineal ideal.

Código ausente. La salida en escalera de la Figura 13.31(a) indica que el código 1001 no aparece en la salida del ADC. Obsérvese que el valor 1000 permanece durante dos intervalos y que la salida salta al valor 1010.

En el convertidor flash, por ejemplo, un fallo en un comparador puede ser la causa del error de omisión de código.



(a) Omisión de código (b) Códigos incorrectos (c) Offset

FIGURA 13.31
Errores de la conversión A/D.

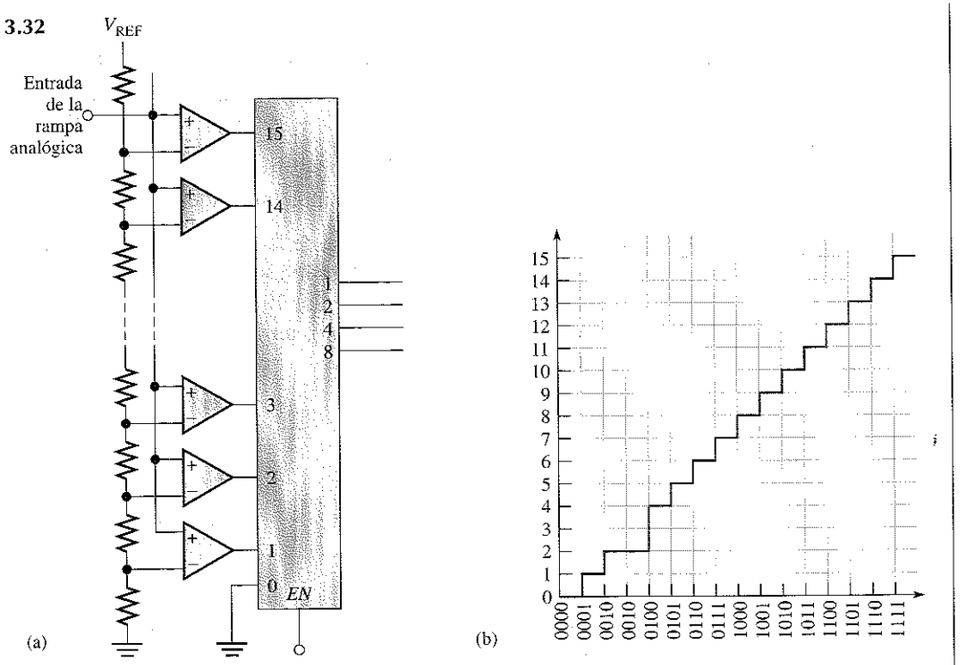
Códigos incorrectos. La salida en escalera de la Figura 13.31(b) indica que varias de las palabras en código binario que salen del ADC son incorrectas. Un análisis indica que, en este caso concreto, la línea del bit 2^1 permanece en estado BAJO (0).

Offset. En la Figura 13.31(c) se presenta la condición de offset. En esta situación, el ADC interpreta que la tensión analógica de entrada es mayor que su valor real. Probablemente, este error se debe a un fallo del circuito comparador.

EJEMPLO 13.5 En la Figura 13.32(a) se presenta un ADC flash de 4 bits, el cual se prueba con un montaje como el mostrado en la Figura 13.30. En la Figura 13.32(b) se muestra la señal analógica de salida reconstruida. Identificar el problema y el fallo más probable.

Solución. El código binario 0011 está ausente en la salida del ADC, como indica el escalón que falta. Muy probablemente, la salida del comparador 3 permanece en su estado inactivo (nivel BAJO).

FIGURA 13.32



Problema relacionado. Reconstruir la señal de salida con un montaje como el de la Figura 13.30, si el comparador 8 del ADC de la Figura 13.32(a) permanece en el estado de salida ALTO.

REVISIÓN DE LA SECCIÓN 13.4

1. ¿Cómo se detecta un comportamiento no monotónico en un DAC?
2. ¿Qué efecto tiene una baja ganancia en la salida de un DAC?
3. Nombrar dos tipos de errores de salida en un ADC.

13.5 ■ INTERFACES INTERNAS DE LOS SISTEMAS

En el Capítulo 12 se ha introducido el concepto de bus en relación con las memorias. Todos los componentes de un sistema digital dado, tal como una computadora, se interconectan mediante buses, que sirven como rutas de comunicación. Físicamente, un bus es un conjunto de conductores que sirven para interconectar dos o más componentes funcionales de un sistema o varios sistemas. Eléctricamente, un bus es una colección de señales y de niveles de tensión y/o de corriente específicos que permiten a los distintos dispositivos conectados al bus trabajar juntos correctamente.

Al finalizar esta sección, el lector deberá ser capaz de:

- Describir el concepto de bus multiplexado.
- Explicar el objeto de las salidas triestado.

Buses multiplexados básicos

Muchos sistemas digitales, como las computadoras, están basados en microprocesadores. En tales sistemas, el microprocesador controla y se comunica con otros dispositivos, tales como memorias y dispositivos de entrada/salida (E/S), por medido de una *estructura interna de buses*, como se indica en la Figura 13.33.

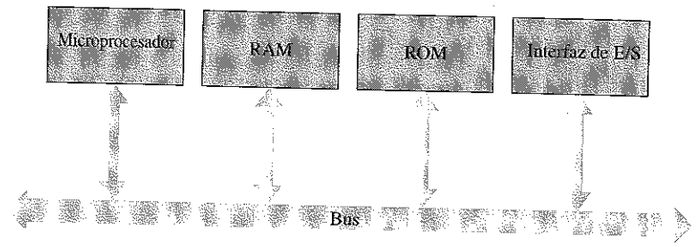


FIGURA 13.33
Interconexión de los componentes de un sistema basado en microprocesador mediante un bus multiplexado bidireccional.

Un **bus** se mutliplexa para que cualquiera de los dispositivos conectados a él pueda enviar o recibir datos desde o hacia uno de los demás dispositivos. A menudo, un dispositivo que envía datos se denomina **fuente** y uno que recibe datos se denomina **aceptor**. En cualquier instante determinado, sólo hay una fuente activa. Por ejemplo, la RAM puede enviar datos a la interfaz de entrada/salida (E/S) bajo el control del microprocesador.

Señales del bus

Con el control síncrono del bus, el microprocesador usualmente genera todas las señales de temporización y control. Luego, los demás dispositivos sincronizan sus operaciones con las señales de temporización y de control. Con el control asíncrono del bus, las señales de temporización y de control son generadas conjuntamente por una fuente y un aceptor. El proceso de establecimiento conjunto de la comunicación se denomina **negociación** (handshaking). En la Figura 13.34 se muestra un sencillo ejemplo de una secuencia de negociación.

Una importante función de control es el **arbitraje del bus**. El arbitraje evita que dos fuentes intenten utilizar el bus al mismo tiempo.

Conexión de dispositivos a un bus

Normalmente, se emplean excitadores **triestado** (consulte el Capítulo 15) para conectar las salidas de un dispositivo fuente a un bus. Usualmente, hay más de una fuente conectada al bus, pero sólo una de ellas puede acceder en cada instante determinado. Las restantes fuentes deben estar desconectadas del bus para evitar una **contienda de bus**.

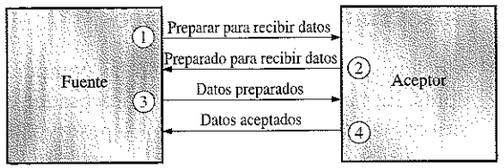


FIGURA 13.34
Ejemplo de una secuencia de establecimiento de comunicación.

Los circuitos triestado se usan para conectar una fuente al bus o para desconectarla, como se muestra en la Figura 13.35(a), en el caso de dos fuentes. La entrada de selección se utiliza para conectar la fuente A o la fuente B al bus, pero no ambas a un mismo tiempo. Cuando la entrada de selección se pone a nivel BAJO, la fuente A se conecta y la fuente B se desconecta. Cuando la entrada de selección se pone a nivel ALTO, la fuente B se conecta y la fuente A se desconecta. En la parte (b) de la figura se muestra el equivalente mediante interruptores de esta operación.

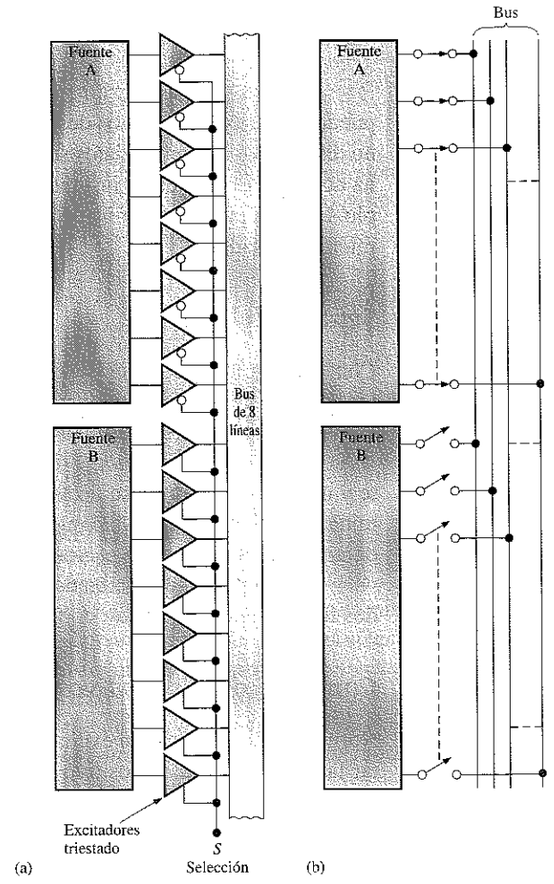


FIGURA 13.35
Drives triestado que implementan la interfaz con un bus.

Cuando la entrada de habilitación de un circuito de tres estados no se activa, el dispositivo se encuentra en estado de alta impedancia (**alta-Z**) y actúa como un interruptor abierto. Muchos circuitos integrados digitales proporcionan excitadores de tres estados internos para las líneas de salida. Una salida de tres estados se indica mediante un símbolo ∇, como muestra la Figura 13.36.

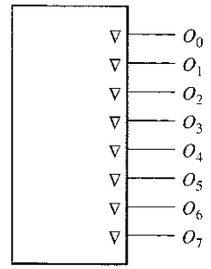


FIGURA 13.36
Método para indicar salidas de tres estados en un circuito integrado.

Funcionamiento del buffer triestado. La Figura 13.37(a) muestra el símbolo lógico para un buffer triestado no inversor con una entrada de habilitación activa a nivel ALTO. La parte (b) de la misma figura muestra el símbolo para otro buffer con una entrada de habilitación activa a nivel BAJO.

La operación básica de un buffer triestado se puede analizar en términos de su acción de conmutación, ilustrada en la Figura 13.38. Cuando la entrada de habilitación está activada, la puerta opera como un circuito normal no inversor. Es decir, la salida está a nivel ALTO cuando la entrada está a nivel ALTO, y estará a nivel BAJO cuando la entrada esté a nivel BAJO, como se muestra en las partes (a) y (b) de la figura, respectivamente. Los niveles ALTO y BAJO representan sólo dos de los estados posibles. El buffer opera en su tercer estado cuando la entrada de habilitación no está activada. En este estado, el circuito actúa como un conmutador abierto, y la salida está completamente desconectada de la entrada, como se muestra en la parte (c) de la figura. A este estado se le denomina en ocasiones estado de *alta impedancia* o de *alta Z*. En el Capítulo 15 se proporcionan explicaciones más detalladas acerca de los circuitos triestado.

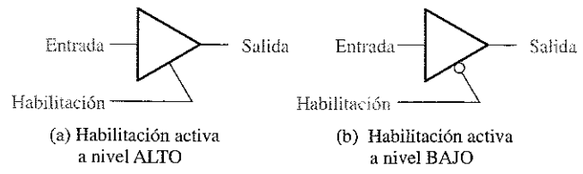


FIGURA 13.37
Símbolos de buffers triestado.

Muchos microprocesadores, memorias y otros circuitos integrados tienen buffers triestado que actúan como interfaz con los buses. Tales buffers son necesarios cuando hay dos o más dispositivos conectados a un bus común. Para evitar que los dispositivos interfieran entre sí, se emplean los buffers triestado para desconectar todos los dispositivos, excepto aquellos que se estén comunicando en cada momento determinado.

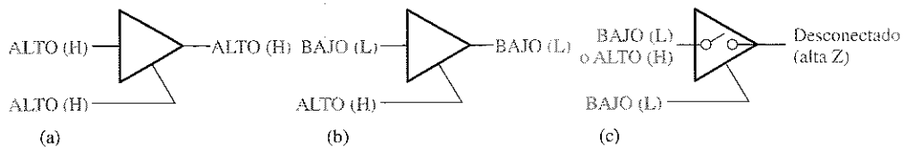


FIGURA 13.38
Operación del buffer triestado.

Contienda de bus

La contienda de bus se produce cuando dos o más dispositivos intentan colocar en la misma línea de bus niveles lógicos opuestos. La forma más común de contienda de bus se produce cuando, antes de que un dispositivo se haya desconectado por completo, otro dispositivo conectado a la línea de bus se activa. Generalmente, esto ocurre en los sistemas de memorias cuando se conmuta del modo lectura al modo escritura o viceversa, y es el resultado de un problema de temporización.

Entradas/salidas multiplexadas

Algunos dispositivos que envían y reciben datos disponen de líneas de entrada y salida combinadas, que se denominan **puertos** de E/S, y que deben multiplexarse en el bus de datos. Los excitadores triestado bidireccionales conectan este tipo de dispositivo con el bus, como se ilustra en la Figura 13.39(a).

Cada puerto de E/S tiene un par de excitadores triestado. Cuando la línea $\overline{SND}/RCV$ (Send/Receive, envío/recepción) está a nivel BAJO, el excitador triestado superior de cada par se activa, y el inferior se desactiva. En este estado, el dispositivo funciona como una fuente y envía los datos al bus. Cuando la línea $\overline{SND}/RCV$ está a nivel ALTO, el excitador triestado inferior de cada par se activa, de modo que el dispositivo funciona como un aceptor y recibe los datos del bus. En la Figura 13.39(b) se ilustra este modo de funcionamiento. Existen algunos dispositivos que realizan la operación de E/S multiplexada mediante circuitería interna.

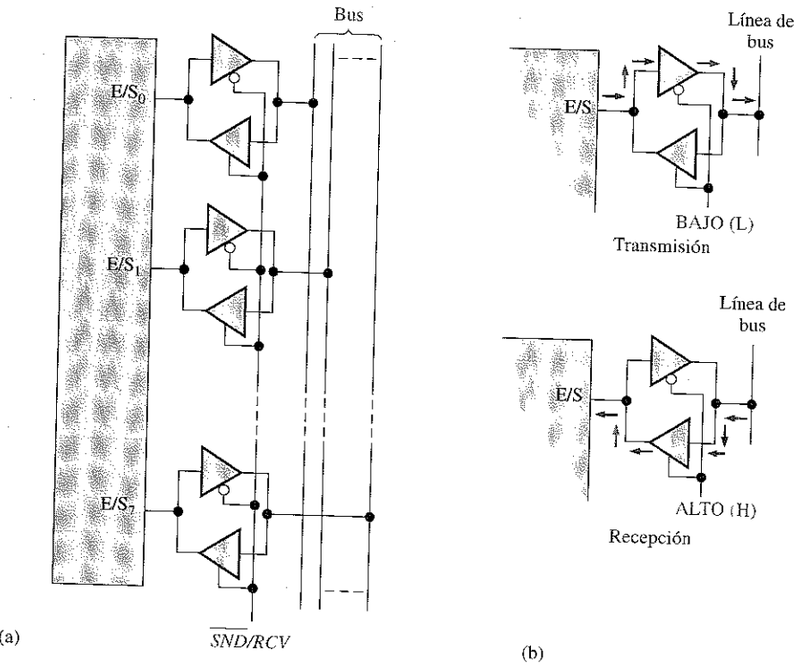


FIGURA 13.39
Operación de E/S multiplexada.

REVISIÓN DE LA SECCIÓN 13.5

1. ¿Por qué se requieren excitadores triestado para realizar la interfaz entre dispositivos digitales y un bus?
2. ¿Cuál es el propósito de un sistema de buses?

13.6 ■ BUSES ESTÁNDAR

Un bus es una "autopista" para las señales digitales; consiste en un conjunto de conexiones físicas (pistas de circuito impreso o cables), por las que se desplazan los datos y otras informaciones desde un lugar a otro. Un bus también consiste en un conjunto estándar de especificaciones que indican las características y tipos de las señales que pueden viajar a su través. Los buses internos interconectan los distintos componentes dentro de un sistema informático: procesador, memoria, unidad de disco, tarjetas controladoras y tarjetas de interfaz. Los buses externos o de E/S permiten transferir señales digitales entre una computadora y el "mundo exterior" y constituyen la interfaz de la computadora con equipos periféricos (monitor de vídeo, teclado, ratón e impresora) o con otros equipos que deban ser controlados mediante una computadora, como puedan ser instrumentos de prueba y medida.

Al finalizar está sección, el lector deberá ser capaz de:

■ Explicar distintos tipos de buses internos y externos, tanto serie como paralelo. ■ Definir el concepto de bus local. ■ Describir los estándares PCI e ISA de bus interno. ■ Describir el bus RS-232C. ■ Describir la interfaz FireWire. ■ Explicar la interfaz USB. ■ Explicar la interfaz GPIB. ■ Explicar la interfaz SCSI.

Buses internos

Los buses internos de una computadora transmiten direcciones, datos y señales de control entre el microprocesador, la memoria caché, la memoria SRAM, la memoria DRAM, las unidades de disco, las ranuras de expansión y otros dispositivos internos. La mayor parte de las computadoras personales de hoy en día poseen tres tipos de buses internos: el *bus local,* el *bus PCI* y el *bus ISA*. La Figura 13.40 muestra la disposición básica de los buses en un sistema.

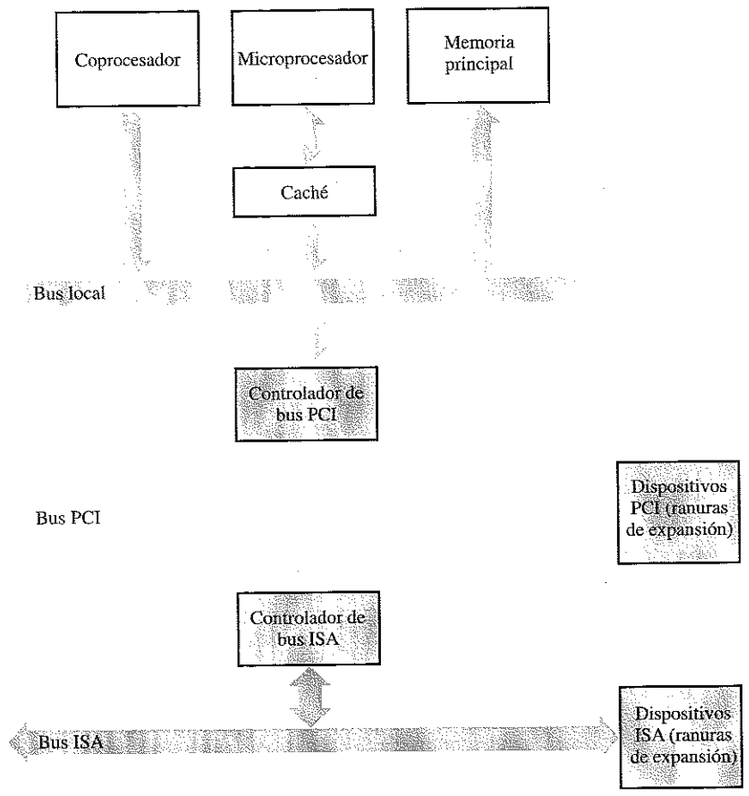


FIGURA 13.40
Ilustración simplificada del sistema básico de buses en una computadora personal típica.

Bus local. Este bus conecta directamente el microprocesador a la memoria caché, a la memoria principal, al coprocesador y al controlador de bus PCI. El **bus local** es el único bus interno que se conecta directamente al microprocesador. Generalmente, este bus incluye los buses de datos, de direcciones y de control que permiten al microprocesador comunicarse con los otros dispositivos. El bus local puede considerarse como el bus *principal* en un sistema informático. Por ejemplo, el bus local del Pentium consta de 32 líneas de dirección de memoria, 64 líneas de datos y numerosas líneas de control.

Bus PCI (Peripheral Control Interconnect, interconexión de control de periféricos). Este bus sirve para establecer la interfaz entre el microprocesador y una serie de dispositivos externos a través de ranuras de expansión (conectores). La interfaz **PCI** fue desarrollada por Intel y, desde que fue introducida en 1993, se ha convertido en el bus de interfaz estándar para las computadoras personales, desplazando a diversos estándares de bus más antiguos. PCI es un bus de 64 bits, aunque a menudo se lo implementa como un bus de 32 bits en el que los buses de direcciones y de datos están multiplexados. Puede operar a velocidades de reloj de 33 MHz o 66 MHz.

El bus PCI está aislado del bus local mediante una unidad controladora de bus que actúa como "puente" entre los dos buses. El PCI se considera un bus "secundario" y su reloj es independiente del reloj del microprocesador. El PCI puede conectar el microprocesador con dispositivos **periféricos**, como pueda ser un disco duro, a través de tarjetas adaptadoras insertadas en ranuras de expansión. La Figura 13.41 muestra el diagrama de pines de las ranuras utilizadas para las tarjetas PCI, para la especificación de 5 V. Existe también una versión de 3,3 V, para las más modernas tarjetas de expansión CMOS a 3,3 V, así como una versión universal.

La interfaz PCI es compatible con las características "plug-and-play", que son la capacidad que tienen ciertas computadoras para configurar de forma automática las tarjetas de expansión y otros dispositivos. Esto permite conectar un dispositivo a una computadora sin preocuparse de configurar conmutadores, cambiar los puentes ni manejar ningún otro elemento de configuración. Esto se consigue mediante una memoria de 256 bytes que permite a la computadora interrogar a la interfaz PCI.

Bus ISA (Industry Standard Architecture, arquitectura estándar de la industria). Este bus de expansión fue desarrollado por IBM para su computadora personal AT y es el bus estándar en el que se insertan casi todas las tarjetas de circuito impreso fabricadas antes de 1993. La interfaz **ISA** se suele incorporar en las computadoras personales más recientes como complemento del bus PCI, para propósitos de compatibilidad descendente. Al final, PCI probablemente acabe reemplazando por completo a la interfaz ISA.

La interfaz ISA tiene un bus de datos de 8 o 16 bits y puede operar a 8,33 MHz. Una versión ampliada denominada EISA proporciona un bus de datos de 32 bits, pero se ha dejado prácticamente de utilizar, debido a su baja velocidad, habiendo sido remplazada por el bus PCI. La ranura ISA está, en realidad, compuesta por un doble conector, como se muestra en la Figura 13.42. El conector superior se utiliza para tarjetas de 8 bits, mientras que la combinación de los dos conectores proporciona la capacidad de transferencia de datos a 16 bits.

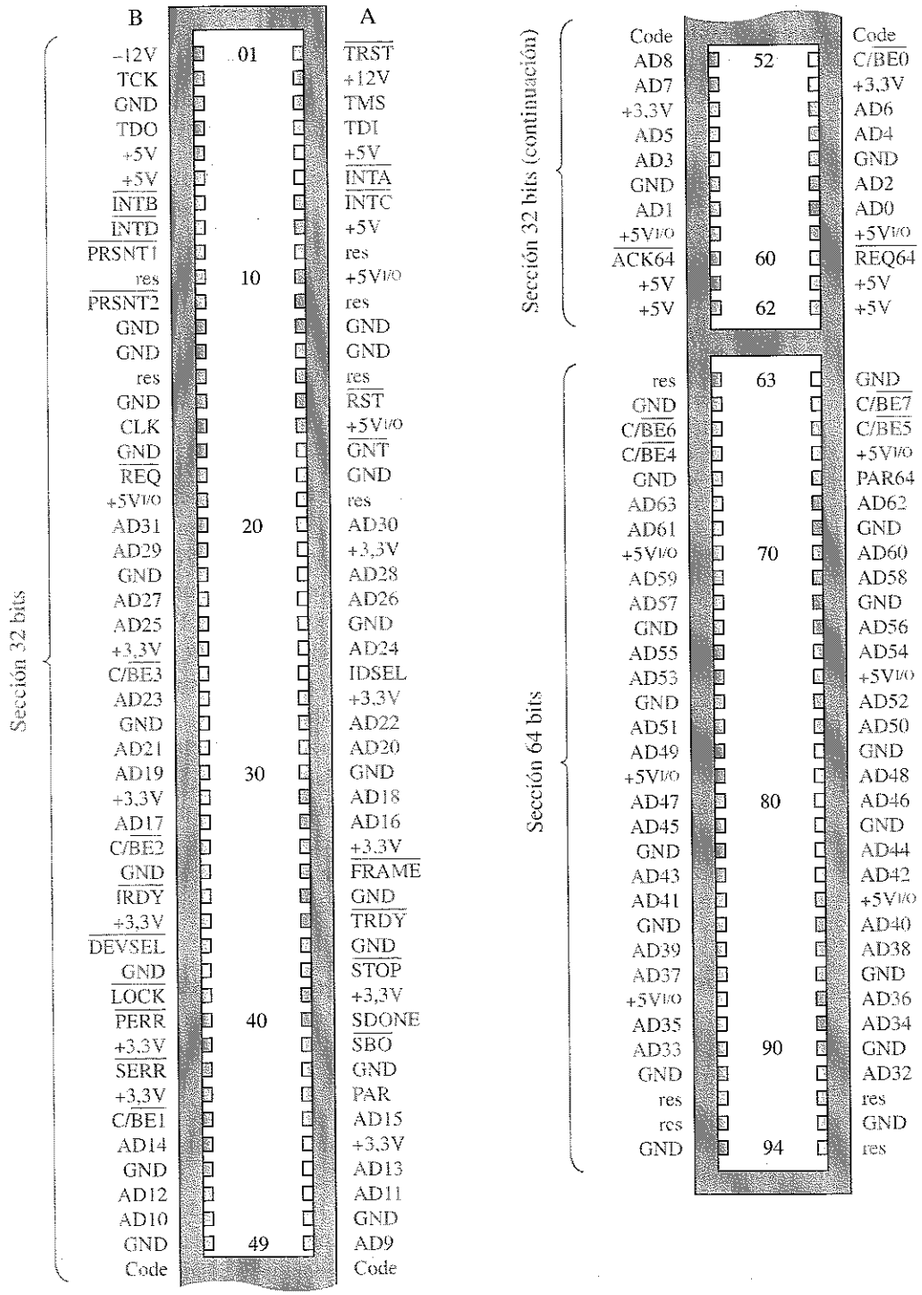
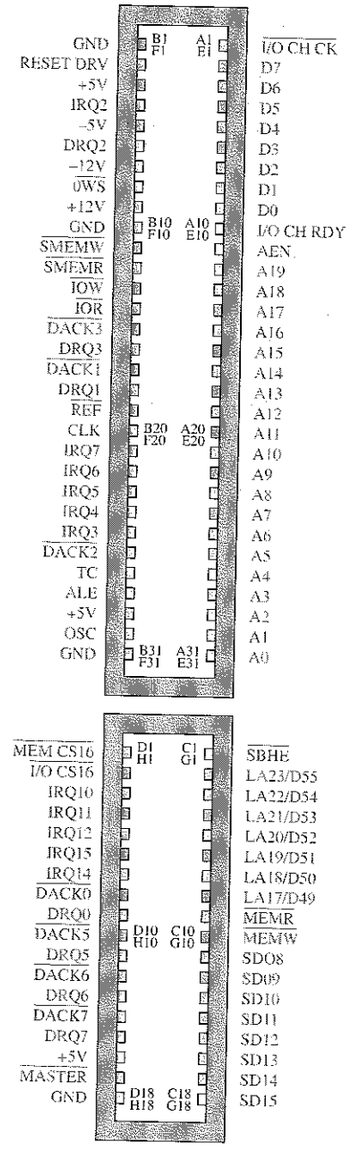


FIGURA 13.41
Diagrama de pines de la ranura de conexión PCI a 5 V utilizada para insertar tarjetas de interfaz. Los pines multiplexados del bus de datos y del bus de direcciones muestran como prefijo las letras AD.

FIGURA 13.42
Diagrama de pines para la ranura de conexión ISA.



Buses externos

Los dispositivos externos se conectan a una computadora mediante una interfaz de entrada/salida (E/S) denominada *puerto*. Existen dos tipos básicos de puertos en una computadora, el *puerto serie* y el *puerto paralelo*, y la mayoría de las computadoras tienen un puerto paralelo y, al menos, un puerto serie para conectar modems, impresoras, ratones y otros dispositivos periféricos.

Un puerto serie se usa para la comunicación de datos serie, donde sólo se transfiere un bit cada vez. Los modems y los ratones son ejemplos de dispositivos serie

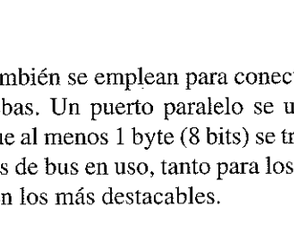
típicos. Algunas veces, los puertos serie también se emplean para conectar la computadora con los equipos de medida y pruebas. Un puerto paralelo se utiliza para la comunicación de datos en paralelo, en la que al menos 1 byte (8 bits) se transfiere cada vez. Actualmente, existen varios estándares de bus en uso, tanto para los puertos serie como paralelo. A continuación se describen los más destacables.

Buses serie para interfaz de E/S

RS-232C. Aprobado por la EIA (Electronic Industries Association) es uno de los estándares más antiguos y más comunes para establecer interfaces serie. El estándar RS-232C también se denomina EIA-232. La mayoría de los modems cumplen el estándar EIA-232, y la mayor parte de las computadoras personales disponen de un puerto RS-232C. El ratón y algunos monitores e impresoras serie, además de los modems, se diseñan para conectarse al puerto RS-232C. El estándar RS-232C se usa habitualmente para establecer la interfaz entre un equipo terminal de datos, **DTE** (Data Terminal Equipment) y un equipo de comunicación de datos, **DCE** (Data Communication Equipment). Por ejemplo, una computadora se clasifica como un DTE y un modem como un DCE.

El estándar EIA-232 especifica veinticinco líneas de conexión entre un DTE y un DCE que requieren un conector de veinticinco pines (DB-25), como se muestra en la Figura 13.43. En las aplicaciones de computadoras personales, no se requieren todas las señales RS-232C. Normalmente, se emplea un mínimo de tres y un máximo de nueve. Por esta razón, IBM definió un conector de 9 pines (DB-9) para su interfaz serie.

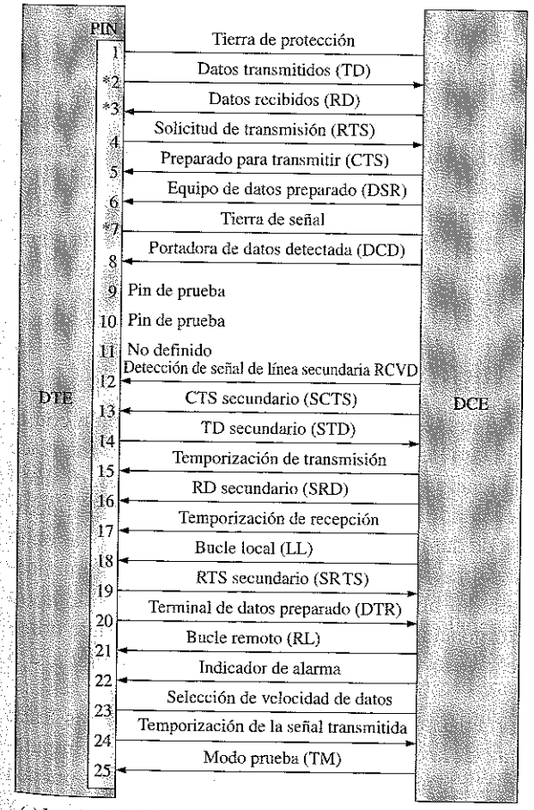
FIGURA 13.43
Conector de 25 pines RS-232C.



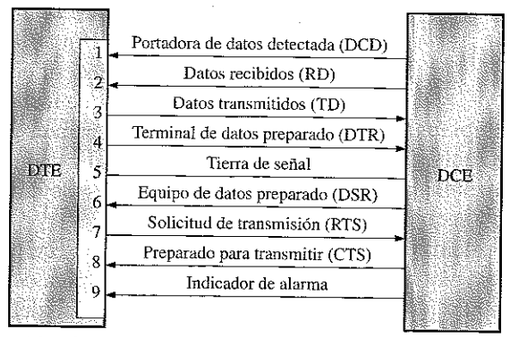
La Figura 13.44(a) enumera las señales y la asignación de pines para el conector RS-232C de 25 pines y, en la parte (b) de la figura, se enumeran las señales y la asignación de pines correspondientes al conector de 9 pines. Los once pines y señales marcadas en color gris claro en la parte (a) indican las señales típicamente utilizadas en las aplicaciones de computadoras personales. Las tres señales mínimas se han marcado mediante un asterisco (pines 2, 3 y 7).

La longitud máxima de cable especificada para la conexión RS-232C es de aproximadamente 150 metros, con una velocidad de transferencia de datos de 20 kbaudios. Si se utiliza un cable más corto, la velocidad en baudios puede ser mayor. La especificación de la velocidad de transferencia de datos en baudios y en bits por segundo (bps) no tienen necesariamente por qué coincidir. La razón de ello es que la velocidad en baudios corresponde a la terminología empleada para los modems y se define como el número de cambios de la señal por segundo, lo que denomina velocidad de modulación. En los modems, algunas veces un cambio de señal transfiere varios bits de datos. Para velocidades bajas, la tasa de transferencias en baudios es igual a la medida de bits por segundo pero, a velocidades altas, la tasa de transferencia en baudios puede ser menor que dicha medida.

Para superar las limitaciones de la interfaz RS-232C, se desarrollaron otros dos estándares: el RS-422 y el RS-423. Estos nuevos estándares especifican longitudes de cable mucho más largas y velocidades de transferencia de datos más altas bajo determinadas condiciones. Por ejemplo, tanto el estándar RS-422 como el RS-423 especifican una longitud máxima de cable de aproximadamente 1.200 metros. La velocidad de transferencia de datos máxima RS-422 es de 10 Mbaudios para un cable de 12 metros y 100 kbaudios para 1.200 metros. Para el estándar RS-423, la velocidad de transferencia de datos es de 100 kbaudios para 90 metros y 1 kbaudio para 1.200 metros. La interfaz RS-232C continúa actualmente siendo la más común.



(a) Interfaz de 25 pines completa RS-232C con la configuración típica de computadora personal indicada en gris claro y tres señales marcadas con asterisco (pines 2, 3 y 7).



(b) Interfaz RS-232C de 9 pines.

FIGURA 13.44
Asignación de pines y señales para ambas versiones de conector RS-232C.

IEEE 1394. Es un estándar de bus serie externo relativamente nuevo y muy rápido, que admite velocidades de transferencia de hasta 400 Mbps y, típicamente, aunque no exclusivamente, se emplea para conectar con los periféricos de vídeo y gráficos, como las cámaras digitales. El estándar **IEEE 1394** a menudo se denomina *FireWire*, un nombre registrado por Apple, que fue quien primero lo desarrolló. Otras empresas utilizan otros nombres para describir sus productos IEEE 1394. IEEE son las siglas de *Institute of Electrical and Electronics Engineers* (Instituto de ingenieros eléctricos y electrónicos).

Se pueden conectar hasta 63 dispositivos a un bus FireWire, con una tecnología de cadena (daisy chain). El cable FireWire consta de seis conductores, dos pares trenzados para datos y dos para la alimentación. Este estándar también admite la posibilidad de "conexión en caliente", es decir, la capacidad de añadir o eliminar de dispositivos conectados a una computadora mientras ésta está en funcionamiento.

USB (Universal Serial bus, bus serie universal). Este estándar de bus serie externo relativamente nuevo puede llegar a reemplazar a la mayoría de los estándares de bus serie y paralelo de velocidad media-baja existentes; quizá la excepción sea el estándar IEEE 1394, que se diseñó para ocupar un hueco distinto en las aplicaciones de alta velocidad y jugar un papel complementario de USB. El **USB** proporciona dos velocidades de transferencia de datos, una alta velocidad de 12 Mbps y una baja velocidad de 1,5 Mbps. Se puede emplear un puerto USB para conectar hasta 127 dispositivos periféricos y el estándar admite tanto características plug-and-play como de conexión en caliente. El cable USB consta de cuatro conductores, dos para datos y dos para alimentación, y conecta la computadora a los dispositivos periféricos USB, pudiendo actuar cualquiera de ellos como concentrador (*hub*) para establecer la conexión con otros dispositivos periféricos USB. La Figura 13.45 ilustra un sistema de computadora con interfaz USB.

Buses paralelo para interfaz de E/S

IEEE 488. Este estándar de bus se emplea hace tiempo y se conoce también con el nombre de bus de interfaz de propósito general (**GPIB,** General Purpose Interface Bus). Ampliamente utilizado en aplicaciones de medida y pruebas, fue desarrollado por Hewlett-Packard en los años sesenta. El estándar **IEEE 488** especifica 24 líneas, que se usan para transferir ocho bits de datos en paralelo a la vez y proporcionar ocho señales de control, que incluyen tres líneas para el establecimiento de la comunicación y cinco líneas para el gobierno del bus. También incluye ocho líneas de tierra para apantallamiento y para los retornos a masa. Actualmente, la velocidad de transferencia de datos es de 1 Mbyte/s y se tiene el propósito de incrementarla a 8 Mbytes/s.

Para conectar un equipo de pruebas a una computadora utilizando el bus IEEE 488, se instala una tarjeta de interfaz en la computadora, que convierte a la computadora en un **controlador** del sistema. En una configuración GPIB típica, se pueden conectar al controlador del sistema hasta 14 dispositivos (instrumentos de medida y pruebas). Cuando el controlador del sistema emite un comando dirigido a dispositivo controlado, con el fin de que éste lleve a cabo una operación específica, como una medida de frecuencia, se dice que el controlador "transmite" y que el dispositivo controlado "recibe".

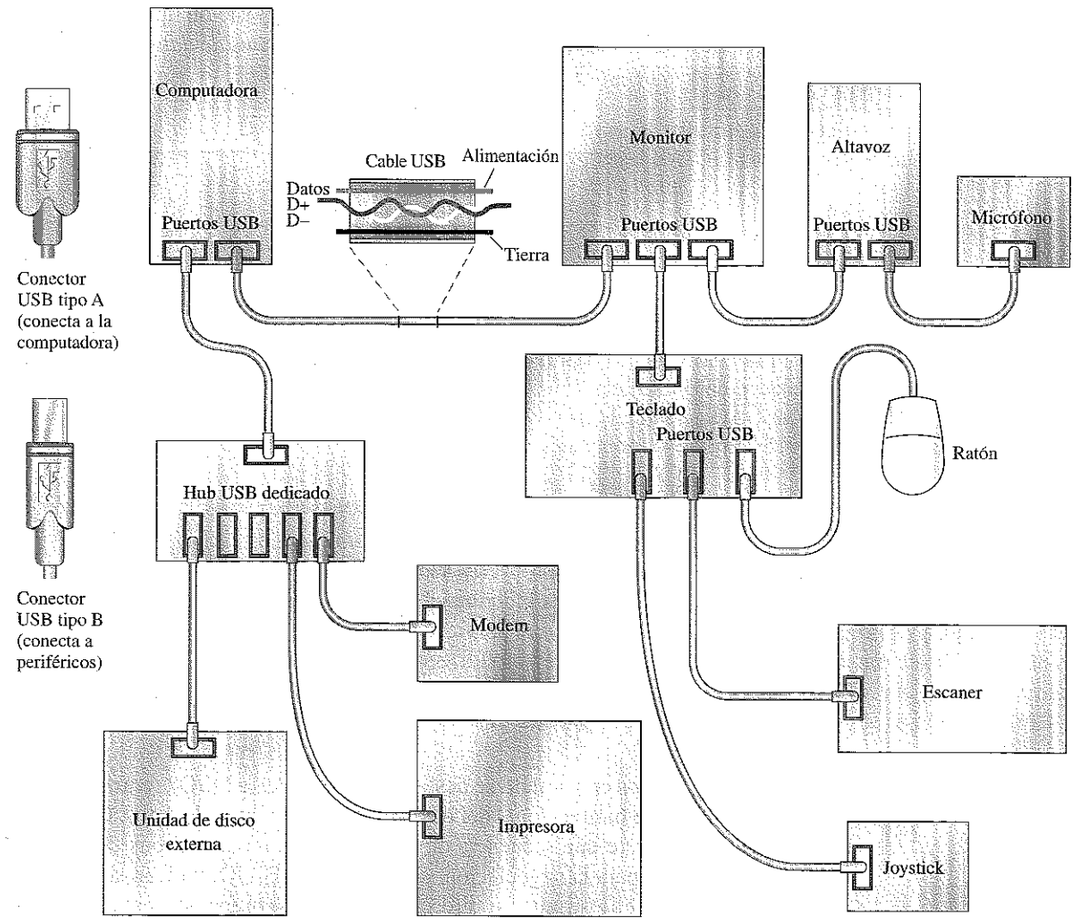


FIGURA 13.45
Ejemplo de un sistema de computadora con interfaz USB.

Un **receptor** es un instrumento capaz de recibir datos en una configuración GPIB cuando el controlador del sistema (computadora) se dirige a él. Ejemplos de receptores son las impresoras, monitores, fuentes de alimentación programables y generadores de señal programables. Un **transmisor** es un instrumento capaz de enviar datos a través del bus GPIB. Ejemplos de transmisores son los multímetros digitales y los contadores de frecuencia que pueden generar datos compatibles con el bus. Algunos instrumentos pueden enviar y recibir datos y se denominan transmisores/receptores; ejemplos de ellos son las computadoras, modems y ciertos instrumentos de medida. El controlador del sistema puede definir a cada uno de los otros instrumentos que hay conectados al bus como receptores o transmisores, de cara a la transferencia de datos. Normalmente, el controlador es un transmisor/receptor.

En la Figura 13.46 se presenta, como ejemplo, una disposición típica GPIB. Las tres agrupaciones básicas de señal de bus se muestran como *bus de datos, bus de control de transferencia de datos* y *bus de gobierno de la interfaz.*

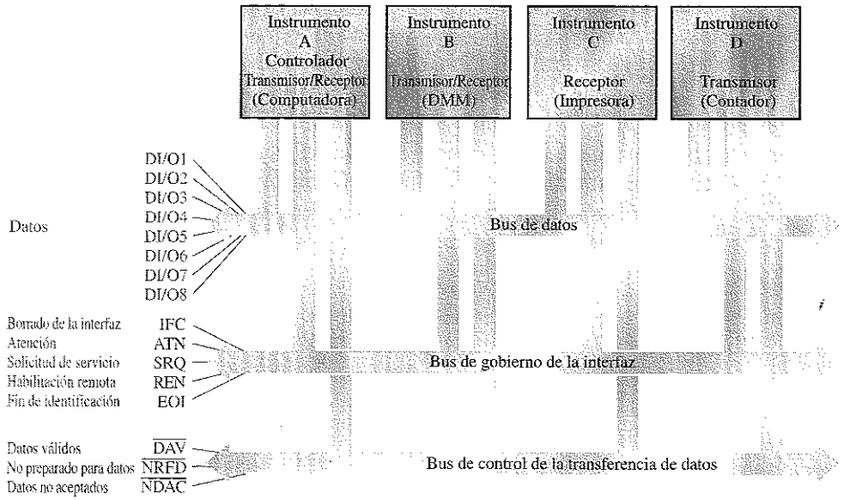


FIGURA 13.46
Una conexión típica IEEE 488 (GPIB)

Las líneas de datos en paralelo se designan como DI/O1a DI/O8 (entrada/salida de datos). En esta parte bidireccional del bus se transfiere un byte de datos. Cada byte que se transfiere requiere una secuencia de negociación a través del bus de transferencia. Las tres líneas para la negociación, activas a nivel BAJO, indican si los datos son válidos ($\overline{\text{DAV}}$), si el instrumento direccionado no está preparado para los datos ($\overline{\text{NRFD}}$) o si los datos no son aceptados ($\overline{\text{NDAC}}$). Más de un instrumento puede recibir datos a un mismo tiempo, y es el instrumento más lento el que establece la velocidad de transferencia. La figura 13.47 muestra el diagrama de tiempos para la secuencia de establecimiento de comunicación GPIB, y la Tabla 13.1 describe las señales para el establecimiento de la comunicación.

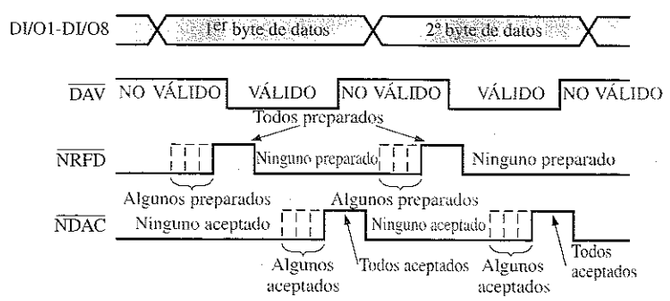


FIGURA 13.47
Cronograma de la secuencia de establecimiento de la comunicación GPIB.

TABLA 13.1
Señales para el establecimiento de la comunicación del GPIB.

Nombre	Descripción
$\overline{\text{DAV}}$	**Data Valid:** Después de que el transmisor detecta un nivel alto en la línea $\overline{\text{NFRD}}$, pone esta línea a nivel BAJO cuando los datos en sus líneas de E/S han sido establecidos y son válidos.
$\overline{\text{NRFD}}$	**Not Ready for Data:** El receptor pone esta línea a nivel bajo para indicar que no está preparado para recibir datos. Un nivel ALTO indica que sí está preparado. La línea $\overline{\text{NFRD}}$ no pasará a nivel ALTO hasta que todos los receptores direccionados estén preparados para recibir datos.
$\overline{\text{NDAC}}$	**Not Data Accepted:** El receptor pone esta línea a nivel BAJO para indicar que no ha aceptado los datos. Cuando acepta datos de sus líneas de E/S, desactiva su línea $\overline{\text{NDAC}}$. La línea $\overline{\text{NDAC}}$ de entrada al transmisor no pasa a nivel ALTO hasta que todos los receptores han aceptado los datos.

Las cinco señales del bus de gobierno de la interfaz controlan el flujo ordenado de los datos. La línea ATN (attention) es supervisada por todos los instrumentos conectados al bus. Cuando la línea ATN está activa, el controlador selecciona la operación de interfaz específica, designa a los transmisores y receptores y proporciona un direccionamiento específico para los receptores. Cada instrumento diseñado según el estándar GPIB tiene una dirección de identificación específica, usada por el controlador del sistema. La Tabla 13.2 describe las líneas de gobierno de la interfaz GPIB y sus funciones.

TABLA 13.2
Líneas de gobierno del bus GPIB.

Nombre	Descripción
ATN	**Attention:** Hace que todos los dispositivos conectados al bus interpreten los datos como una dirección o un comando del controlador, y activa la función de establecimiento de la comunicación.
IFC	**Interface Clear:** Inicializa el bus
SRQ	**Service Request:** Alerta al controlador de que un dispositivo necesita comunicarse.
REN	**Remote Enable:** Habilita a los dispositivos para responder a un control remoto de programación.
EOI	**End or Identify:** Indica el último byte de datos que se transfiere.

El GPIB está limitado a una longitud máxima de cable de 15 metros y no puede haber más de un dispositivo por metro, con una capacidad de carga máxima de 50 pF por dispositivo. La limitación de la longitud de cable se puede superar utilizando modems y extensores de bus. Un extensor de bus permite conectar el cable de interfaz

a los instrumentos que están separados por una distancia mayor que la permitida por el GPIB, o comunicarse a larga distancia vía modem-línea telefónica. En la Figura 13.48 se ilustra el uso de los extensores de bus y/o modems.

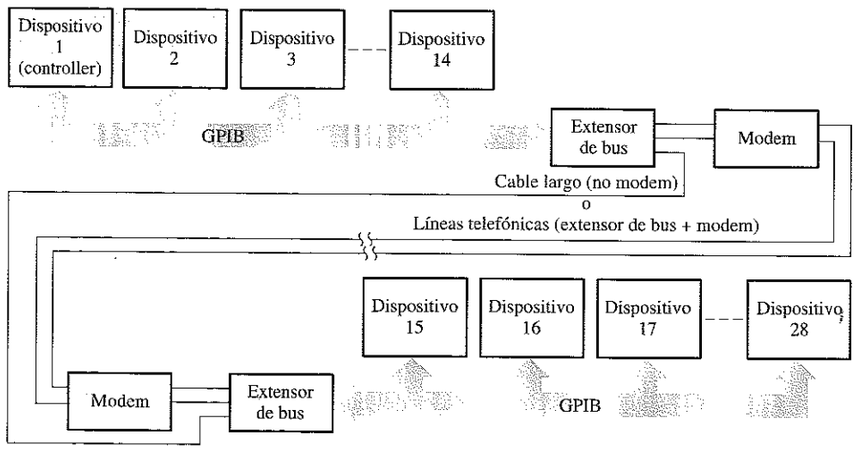


FIGURA 13.48
Se puede utilizar un extensor de bus y modem para establecer la interfaz con sistemas GPIB remotos.

SCSI (Small Computer System Interface, interfaz para sistemas informáticos de pequeño tamaño). Se trata de un estándar ampliamente utilizado para conectar computadoras personales y periféricos. Aunque **SCSI** es un estándar ANSI (American National Standards Institute, Instituto nacional de estándares de EEUU), existen diversas variaciones y tipos de conectores, de distintos fabricantes. Puede ocurrir que un tipo de SCSI no sea compatible con otro tipo. SCSI-1 es la versión con conector de 25 pines que proporciona un bus de datos de 8 bits y admite velocidades de transferencia de datos de 4 Mbytes/s. A continuación se enumeran otras versiones del estándar de bus SCSI:

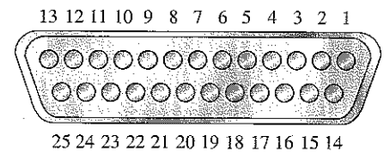
- *SCSI-2*. Esta versión es la misma que SCSI-1, pero utiliza un conector de 50 pines y admite múltiples dispositivos.
- *Wide SCSI* (SCSI ancho). Utiliza un conector más ancho que el tipo SCSI-2, para admitir transferencias de datos de 16 bits.
- *Fast SCSI* (SCSI rápido). Proporciona transferencia de datos de 8 bits, pero admite velocidades de transferencia de datos de 10 Mbytes/s.
- *Fast Wide SCSI*. Esta versión permite la transferencia de datos de 16 bits a 20 Mbytes/s.
- *Ultra SCSI*. Esta versión transfiere 8 bits de datos a 20 Mbytes/s.
- *SCSI-3*. Esta versión tiene 16 líneas de datos y funciona a 40 Mbytes/s.
- *Ultra SCSI-2*. Esta versión transfiere 8 bits a 40 Mbytes/s.
- *Wide Ultra SCSI-2*. Esta versión proporciona transferencias de datos de 16 bits y opera a 80 Mbytes/s.

Las descripciones de señal para un conector de 25 pines SCSI se proporcionan en la Tabla 13.3, mientras que la configuración de pines se muestra en la Figura 13.49.

TABLA 13.3
Señales SCSI.

Número de pin	Nombre de la señal	Descripción de la señal
1	REQ/	Request (Solicitud)
2	MSG/	Message (Mensaje)
3	I/O/	Input/Output (Entrada/salida)
4	RST/	SCSI bus reset (Puesta a cero del bus SCSI)
5	ACK/	Acknowledge (Confirmación)
6	BSY/	Busy (Ocupado)
7	GND	Señal de tierra
8	DB0/	Bit de datos 0
9	GND	Señal de tierra
10	DB3/	Bit de datos 3
11	DB5/	Bit de datos 5
12	DB6/	Bit de datos 6
13	DB7/	Bit de datos 7
14	GND	Señal de tierra
15	C/D/	Command/Data (Comando/datos)
16	GND	Señal de tierra
17	ATN/	Attention (Atención)
18	GND	Señal de tierra
19	SEL/	Select (Selección)
20	DBP/	Paridad de datos
21	DB1/	Bit de datos 1
22	DB2/	Bit de datos 2
23	DB4/	Bit de datos 4
24	GND	Señal de tierra
25	TPWR	Alimentación del terminador

FIGURA 13.49
Conector de 25 pines SCSI.



REVISIÓN DE LA SECCIÓN 13.6

1. Nombrar las dos principales categorías de buses en términos del método de transferencia de datos.
2. Clasificar cada uno de los siguientes buses como bus serie o bus paralelo:
   (a) SCSI **(b)** RS-232C **(c)** USB **(d)** GPIB
3. Explicar la diferencia básica entre un bus serie y un bus paralelo.
4. ¿Cuántos dispositivos pueden conectarse al bus USB?
5. ¿Es el bus FireWire más rápido que el USB en términos de transferencia de datos?

13.7 ■ APLICACIÓN A LOS SISTEMAS DIGITALES

En esta aplicación a los sistemas digitales, un ADC se aplica a un sistema de posicionamiento de antena de satélite. Las entradas analógicas que indican la posición de la antena, elevación y azimut, se aplican al ADC, que convierte las posiciones en códigos digitales que se visualizan en un display. La antena se puede ajustar remotamente hasta obtener la posición deseada.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar el funcionamiento básico de un multiplexor analógico. ■ Explicar el funcionamiento global del sistema. ■ Comentar el uso del ADC en esta aplicación concreta. ■ Desarrollar un procedimiento de prueba.

Funcionamiento general del sistema

Se requieren dos posiciones angulares para dirigir correctamente la antena al satélite deseado. El azimut es un arco de este a oeste (o de norte a sur) paralelo con el horizonte. La elevación va de horizonte a horizonte formando un arco vertical. Estos dos movimientos angulares de la antena se ilustran en la Figura 13.50. Dos motores controlan la antena para colocarla en la posición adecuada, uno para el azimut y otro para la elevación. También hay dos transductores de posición (en esta aplicación, se usan potenciómetros como sensores angulares), que generan tensiones proporcionales a las direcciones actuales de la antena, determinadas por la posición angular de los ejes de los motores.

En la Figura 13.51 se presenta un diagrama de bloques básico del sistema de posicionamiento de antena. Una de las entradas del multiplexor analógico proviene del potenciómetro del azimut y la otra procede del potenciómetro de elevación. La entrada de selección conmuta alternativamente las dos tensiones analógicas de entrada al ADC, el cual las convierte en un código digital de 8 bits. Durante un intervalo, la tensión analógica que representa la posición actual del azimut se convierte a formato digital y se visualiza en el display. Durante el siguiente intervalo, la tensión analógica correspondiente a la posición actual de la elevación se convierte a formato digital y se visualiza en el display. La posición de la antena se fija mediante los controles de los motores, que pueden situarse alejados de la antena y conectarse a los motores a través de un cable. Observando las lecturas de las posiciones de azimut y elevación, la antena puede llevarse a la posición deseada.

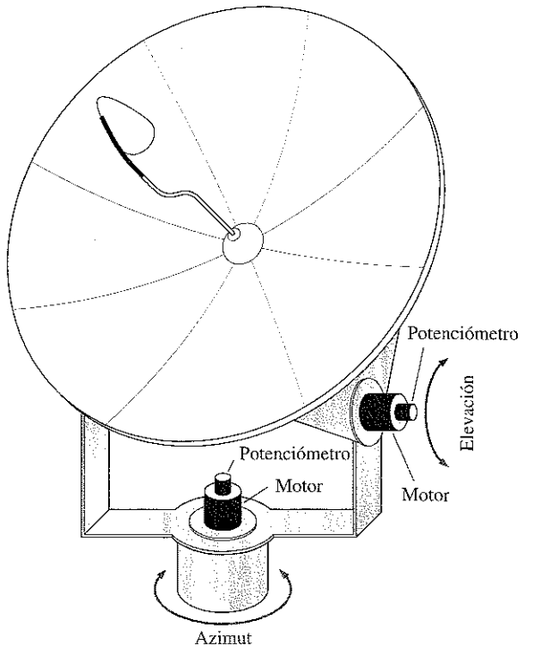
El multiplexor analógico

Facilitamos una descripción general del multiplexor analógico para que pueda familiarizarse con este tipo de dispositivo antes de abordar su aplicación dentro del sistema.

En los sistemas de adquisición de datos en los que las entradas de diferentes fuentes deben convertirse independientemente a un formato digital para su procesamiento, se utiliza una técnica que se denomina *multiplexación*. Se emplea un interruptor analógico separado para cada fuente analógica, como se ilustra en la Figura 13.52 para un sistema de 4 canales. En este tipo de aplicación, todas las salidas de los conmutadores analógicos se conectan a un mismo punto para formar una salida común, y sólo un

conmutador puede estar cerrado en un determinado instante de tiempo. La salida común de los conmutadores se conecta a la entrada de un amplificador operacional seguidor de tensión, como se ve en la figura.

FIGURA 13.50
Plato de antena con controles de posición y sensores para el azimut y la elevación.



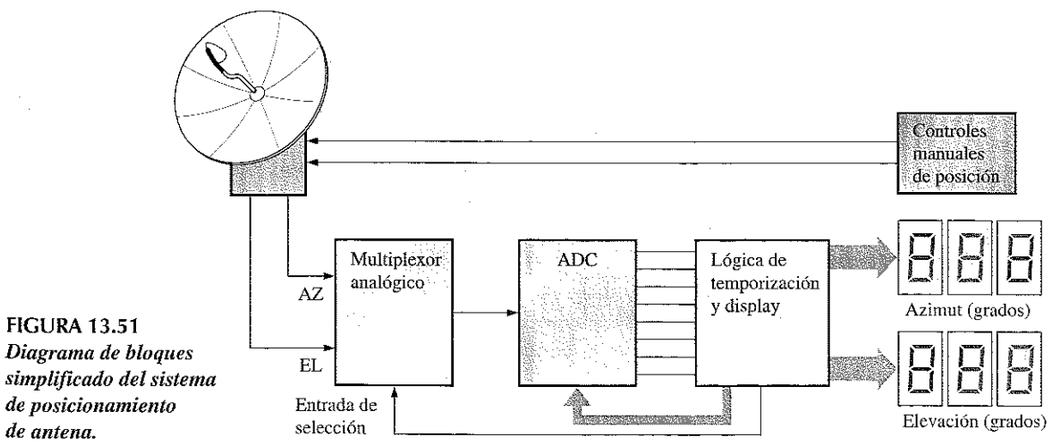


FIGURA 13.51
Diagrama de bloques simplificado del sistema de posicionamiento de antena.

Un buen ejemplo de un circuito integrado multiplexor analógico es el AD9300 presentado en la Figura 13.53. Este dispositivo está formado por cuatro conmutadores

analógicos controlados por un decodificador de canal. Las entradas A_0 y A_1 determinan cuál de los cuatro conmutadores se activa. Si A_0 y A_1 están a nivel BAJO, se selecciona la entrada In_1. Si A_0 está a nivel ALTO y A_1 está a nivel BAJO, se selecciona la entrada In_2. Si A_0 está a nivel BAJO y A_1 está a nivel ALTO, se selecciona la entrada In_3. Si A_0 y A_1 están a nivel ALTO, se selecciona la entrada In_4. La entrada de habilitación (enable) controla el conmutador que conecta o desconecta la salida.

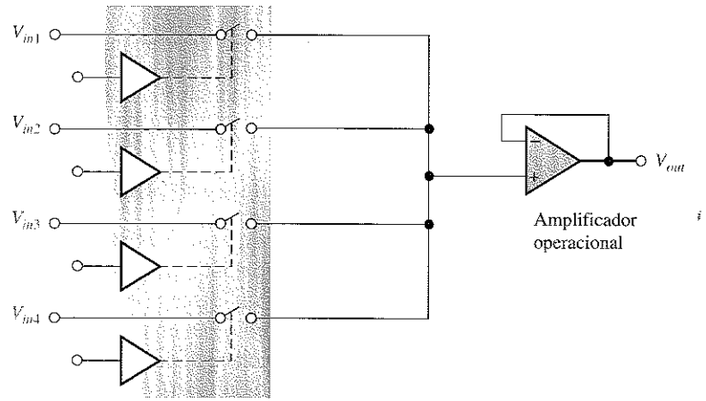


FIGURA 13.52
Multiplexor analógico de 4 canales básico.

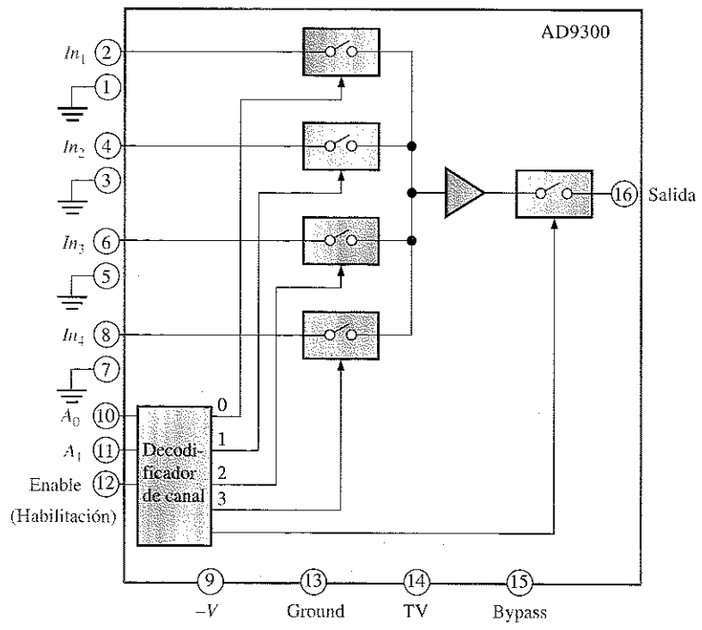


FIGURA 13.53
El multiplexor analógico AD9300.

El potenciómetro

En esta aplicación, el potenciómetro se usa como un transductor de posición, para convertir la posición angular del eje del motor en una tensión continua proporcional a la posición. El potenciómetro está unido mecánicamente al motor asociado de forma que, a medida que gira el eje del motor, el cursor (contacto deslizante) del potenciómetro se desliza a lo largo del elemento resistivo. Éste se calibra para que el menor ángulo genere la menor tensión. En la Figura 13.54 se muestra un diagrama simplificado de la construcción básica y el esquema de la parte (b) indica los ángulos mínimo y máximo correspondientes a la posición del cursor.

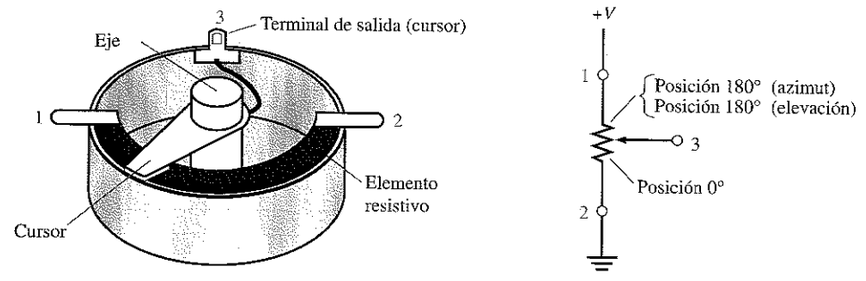


FIGURA 13.54
El potenciómetro como transductor angular.

Funcionamiento básico de los circuitos analógico-digitales

La Figura 13.55 muestra los circuitos analógico-digitales específicos utilizados en el sistema. Las tensiones analógicas procedentes de los potenciómetros de azimut y elevación de la antena se aplican a las dos entradas del multiplexor analógico AD9300. Las otras dos entradas analógicas no se utilizan y se conectan a tierra. La lógica de temporización digital produce una señal cuadrada que se utiliza como entrada de selección, A_0. Cuando la señal cuadrada está a nivel BAJO, se selecciona la entrada 1, In_1, y la tensión de azimut se conmuta a la entrada del ADC. Cuando la señal cuadrada está a nivel ALTO, se selecciona la entrada 2, In_2, y la tensión de elevación se conmuta a la entrada del ADC. Aunque en esta aplicación se utiliza el AD673, se pueden utilizar otros ADC, como por ejemplo el ADC0804.

Cada vez que se presenta una tensión analógica en la entrada del ADC, la lógica de temporización genera un impulso para que el ADC inicie la conversión. Después de que se completa la conversión, el ADC pone un nivel BAJO en la línea de datos preparados dirigida a la lógica de temporización, la cual responde con una confirmación a través de la línea de habilitación de datos, poniéndola a nivel bajo. Después, el ADC coloca el código binario de 8 bits en sus salidas. El cronograma se ilustra en la Figura 13.56.

A medida que las tensiones de azimut y elevación se convierten a código binario de manera alternativa, la lógica del display almacena cada código en su registro

asociado. Después, el código binario se convierte a BCD y luego a formato de 7 segmentos para iluminar el display de tres dígitos. Los displays se actualizan muchas veces por segundo, debido a la operación de multiplexación. La velocidad de actualización depende de la frecuencia de la señal cuadrada utilizada en la entrada de selección del multiplexor.

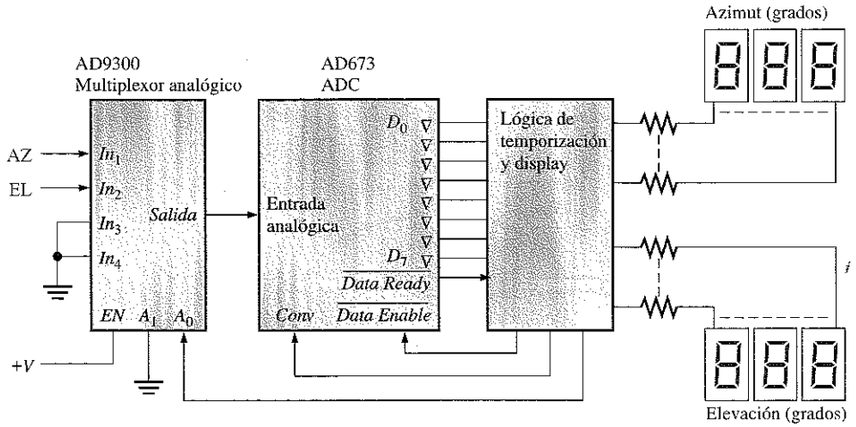


FIGURA 13.55
Circuitos de conversión y lógica de temporización/display del sistema de posicionamiento de antena.

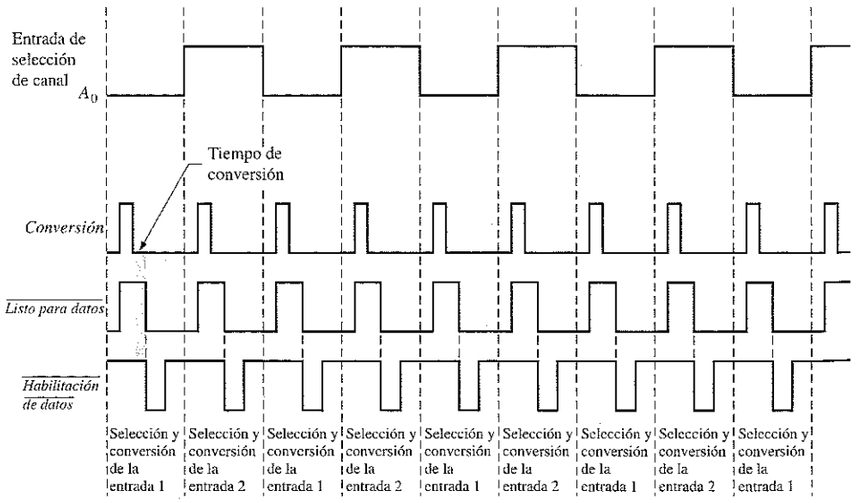


FIGURA 13.56
Cronograma básico del sistema de la Figura 13.55.

■ PRÁCTICAS DE SISTEMAS

El diagrama lógico básico de la Figura 13.55 y el cronograma de la Figura 13.56 indican los requisitos generales del display y de la lógica de temporización, pero no representan de forma detallada la lógica necesaria para implementar el sistema. En los siguientes bancos de trabajo va a desarrollarse, a partir de las especificaciones dadas, la circuitería detallada que se requiere para implementar las funciones del display y de la lógica de temporización del sistema.

■ PRÁCTICA DE SISTEMAS 1: Diseño de la lógica del display

Esta práctica de sistemas se centra en el diseño de la lógica del display, cuyo diagrama de bloques básico se muestra más abajo. Consultar los capítulos anteriores y/o las hojas de especificaciones de los dispositivos si se cree necesario.

- ***Actividad 1. Convertidor de código binario-BCD.***

  El código binario de 8 bits del ADC se convierte en un código BCD de 12 bits (tres dígitos). Especificar un circuito o una combinación de circuitos que realicen esta función.

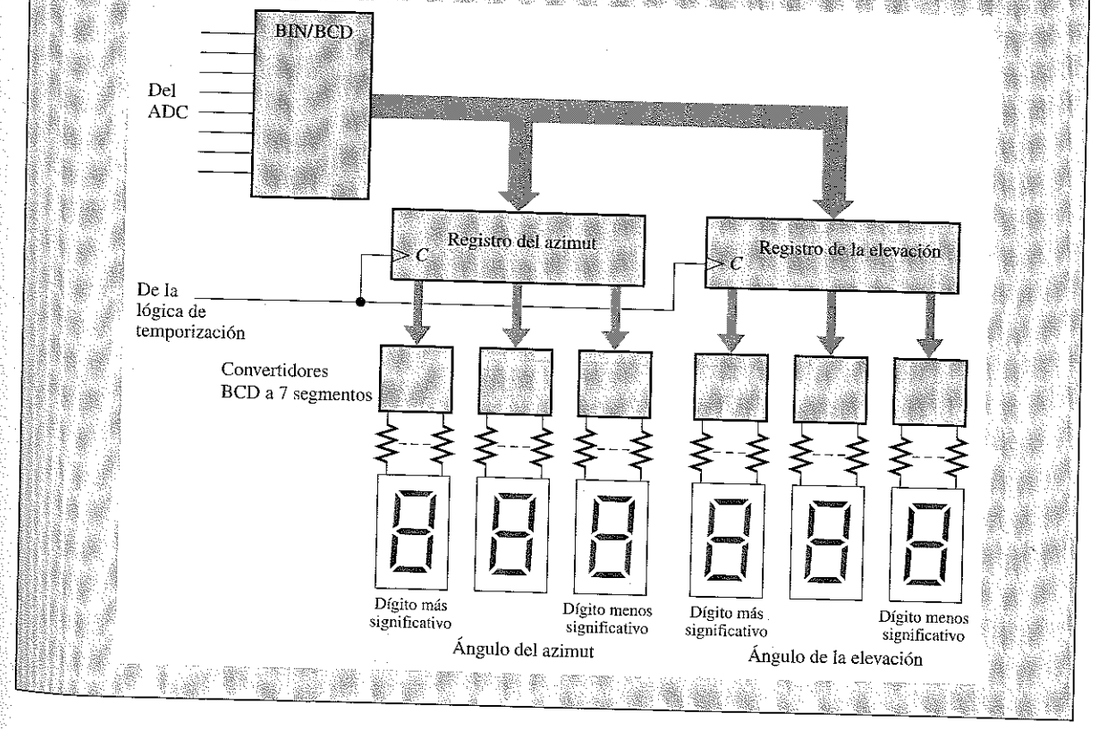
- ***Actividad 2. Registros para el azimut y la elevación.***

  El valor del azimut en código BCD se almacena en un registro y el código BCD correspondiente a la elevación, en otro. Los registros se actualizan de forma alternativa. Obsérvese que se requiere trabajar con entrada y salida paralelo. Especificar circuitos integrados de registro que cumplan estos requisitos.

- ***Actividad 3. Decodificador BCD a 7 segmentos.***

  Se requiere un decodificador de BCD a 7 segmentos para cada uno de los tres dígitos que indican el valor del azimut y para cada uno de los tres dígitos que indican el valor de la elevación. Especificar un dispositivo apropiado. Especificar también un display de 7 segmentos y determinar los valores de resistencia adecuados para establecer la interfaz entre los decodificadores y los displays.

- ***Actividad 4.*** Desarrollar un diagrama lógico detallado que aplique los dispositivos especificados en las actividades anteriores. Mostrar todos los detalles necesarios, incluyendo la numeración de los pines.



■ PRÁCTICA DE SISTEMAS 2: Diseño de la lógica de temporización

Esta práctica de sistemas se centra en el diseño de la lógica de temporización, cuyo diagrama de bloques básico para dos conversiones se muestra más abajo. Consultar los capítulos anteriores y/o las hojas de especificaciones de los dispositivos si se cree necesario.

- ***Actividad 1. Señal de selección de canal***

  Esta señal selecciona alternativamente el canal 1 (In_1) y el canal 2 (In_2) del ADC para la conversión a binario.
  Especificaciones: señal cuadrada, a una frecuencia que proporcione un mínimo de 100 conversiones de cada entrada analógica por segundo. Desarrollar un circuito para generar esta forma de onda cuadrada.

- ***Actividad 2. Señal de conversión***

  Cada impulso de conversión positivo inicia una conversión en el ADC mediante su flanco posterior. El tiempo máximo de conversión del AD673 es 30 μs. Es necesario un impulso para cada conversión. Especificar un circuito que genere esta señal.
  Especificaciones: los impulsos positivos tienen una anchura mínima de 500 ns y están retrasados como mínimo 100 ns con respecto a cada flanco de la señal de selección de canal.

- ***Actividad 3. Señal de habilitación de datos.***

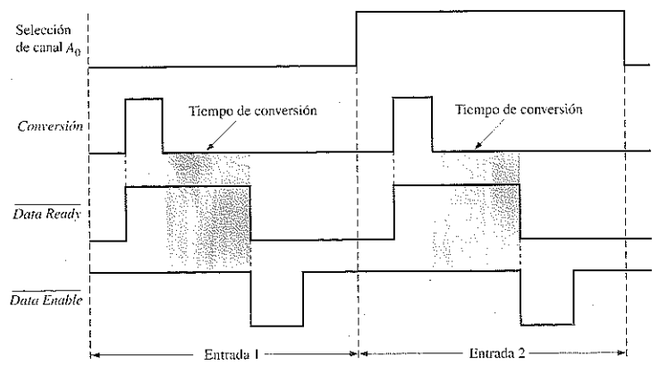
  Cuando el AD673 completa una conversión, pone a nivel BAJO la línea de datos preparados, la cual permanece a nivel BAJO hasta el flanco anterior del siguiente impulso de conversión. La línea de datos preparados debe permanecer a nivel ALTO durante la conversión. Al terminar la conversión, la habilitación de datos pasa a nivel BAJO, para colocar el código binario en las líneas de salida del ADC. Especificar un dispositivo que genere esta señal.
  Especificaciones: Pulsos negativos disparados a partir de las transiciones negativas de la línea de datos preparados; la anchura mínima del impulso debe ser de 500 ns.

- ***Actividad 4. Almacenamiento de los registros de azimut y elevación.***

  Determinar la temporización de la señal de reloj requerida para introducir los datos en paralelo en los registros. Especificar cómo se genera.

- ***Actividad 5.*** Desarrollar un diagrama lógico detallado de la lógica de temporización y combinarlo con el diagrama de la lógica del display desarrollado en la Práctica de sistemas 1.

- ***Actividad 6.*** Escribir un informe técnico que incluya una descripción detallada del funcionamiento del sistema y una lista de componentes.



REVISIÓN DE LA SECCIÓN 13.7

1. ¿Por qué se requieren tres dígitos en los displays en los que se visualizan el azimut y la elevación?
2. Comentar cómo se generan las tensiones analógicas que representan el azimut y la elevación.
3. ¿Por qué se requieren dos registros en la lógica del display?

■ RESUMEN

- Tipos de convertidores digital-analógicos (DAC):
  con ponderación binaria
  en escalera *R/2R* (más común)
- Tipos de convertidores analógico-digitales (ADC):
  flash (el más rápido)
  de rampa en escalera
  de seguimiento
  de pendiente simple
  de pendiente doble
  por aproximaciones sucesivas (el más común).
- La Tabla 13.4 presenta una comparación de los buses estándar

TABLA 13.4

	Buses internos		Buses externos				
	PCI	**ISA**	**RS-232C**	**IEEE 1394**	**USB**	**IEEE 488**	**SCSI**
Tipo	paralelo	Paralelo	serie	serie	serie	paralelo	paralelo
Líneas de datos	32/64	8/16	—	—	—	8	8/16
Velocidad de los datos	33/66 MHz	8.33 MHz	20 kbaud	400 Mbps	1,5/12 Mbps	1 Mbps	4 Mbps (1) 10 Mbps (Fast) 20 Mbps (Ultra) 40 Mbps (3)
Números de dispositivos	—	—	1	63	127	14	16

■ PALABRAS CLAVE

Estos términos también se encuentran en el glosario del final del libro.

Aceptor Dispositivo receptor en un bus.

ADC flash o paralelo Convertidor analógico-digital simultáneo.

Alta-Z Estado de alta impedancia de un circuito triestado, en el que la salida se desconecta del resto del circuito.

Arbitraje del bus El proceso que evita que dos fuentes utilicen un bus al mismo tiempo.

Bus Grupo de líneas digitales que establece la interfaz entre uno o más dispositivos basándose en una especificación estandarizada.

Bus IEEE-488 También denominado GPIB (bus de interfaz de propósito general). Bus paralelo estándar utilizado ampliamente como interfaz de prueba y medida.

Bus IEEE 1349 Bus serie de transferencia de datos a alta velocidad. También conocido como FireWire.

Bus ISA Industry Standard Architecture, arquitectura estándar de la industria. Estándar de bus paralelo interno.

Bus local Bus interno que conecta el microprocesador a la memoria caché, la memoria principal, el coprocesador y el controlador de bus PCI.

Bus PCI Peripheral Control Interconnect, bus de interconexión de control de periféricos. Un estándar de bus paralelo interno.

Contienda de bus Una condición adversa que podría ocurrir si dos o más dispositivos intentan comunicarse a un mismo tiempo a través de un bus.

Controlador Instrumento que puede definir a los demás instrumentos conectados al bus como transmisores o receptores durante una transferencia de datos.

Convertidor analógico-digital (ADC) Dispositivo utilizado para convertir una señal analógica a formato digital.

Convertidor digital-analógico (DAC) Dispositivo utilizado para convertir una entrada digital en una señal analógica.

DCE Data Communications Equipment, equipo de comunicación de datos.

DTE Data Terminal Equipment, equipo terminal de datos.

Establecimiento de la comunicación, negociación (handshaking) Método o proceso de intercambio de señales, mediante el cual dos dispositivos o sistemas digitales establecen conjuntamente una comunicación.

Fuente Dispositivo que envía algo por un bus.

GPIB General Purpose Interface Bus, bus de interfaz de propósito general, basado en el estándar IEEE 488.

Interfaz Mecanismo por el que los dos o más dispositivos electrónicos o sistemas se hacen compatibles operacionalmente entre sí, de manera que puedan funcionar adecuadamente juntos.

Modem Modulador/demodulador que sirve para conectar dispositivos digitales con sistemas de transmisión analógicos, como las líneas telefónicas.

Monotonicidad Característica de un DAC definida por la ausencia de inversiones de paso incorrectas. Un tipo de linealidad digital-analógica.

Periférico Dispositivo o instrumento que proporciona servicios de comunicación con alguna computadora, o proporciona servicios o funciones auxiliares a una computadora.

Puerto Interfaz física entre una computadora y un periférico.

Receptor Instrumento capaz de recibir datos en un bus GPIB (bus de interfaz de propósito general).

SCSI Small Computer System Interface (interfaz para sistemas informáticos de pequeño tamaño). Estándar de bus paralelo externo.

Transmisor Instrumento capaz de transmitir datos por un bus GPIB (bus de interfaz de propósito general).

Triestado Un tipo de salida en un circuito lógico que posee tres estados: ALTO, BAJO y alta-Z.

USB Universal Serial Bus (bus serie universal). Estándar de bus serie externo.

■ AUTOTEST

1. En un convertidor digital-analógico (DAC) con ponderación binaria, las resistencias de entrada
   (a) determinan la amplitud de la señal analógica
   (b) determinan los pesos de las entradas digitales
   (c) limitan el consumo de potencia
   (d) evitan cargar a la fuente
2. Cada uno de los valores de resistencia de un DAC con ponderación binaria es
   (a) igual a las demás resistencias
   (b) la mitad de la resistencia de realimentación
   (c) dos veces el valor de la resistencia inmediata de menor valor
   (d) la mitad del valor de la resistencia inmediata de menor valor
3. En un DAC *R/2R*, hay
   (a) cuatro valores de resistencia
   (b) un valor de resistencia
   (c) dos valores de resistencia
   (d) un numero de valores de resistencia igual al número de entradas
4. Un DAC de 8 bits tiene una resolución de
   (a) 0,1% **(b)** 0,392% **(c)** 1% **(d)** 3,92%
5. Idealmente, la precisión de un DAC de 4 bits es
   (a) aproximadamente el 3,1%
   (b) aproximadamente el 0,31%
   (c) aproximadamente el 6,2 %
   (d) aproximadamente el 0,62%
6. El convertidor analógico-digital (ADC) con un tiempo de conversión mayor es el de tipo:
   (a) flash **(b)** de rampa digital
   (c) de seguimiento **(d)** conversión simultánea
7. La mayoría de los dispositivos realizan la interfaz con un bus mediante
   (a) salidas totem-pole **(b)** excitadores triestado
   (c) transistores *pnp* **(d)** resistencias
8. El bus PCI está formado por
   (a) 8 o 16 líneas de datos
   (b) 32 o 64 líneas de datos
   (c) 1 línea de datos serie
9. Los dispositivos que trabajan con un bus GPIB se denominan
   (a) fuente y carga
   (b) transmisor y receptor
   (c) donador y aceptor

10. RS-232C es

(a) una interfaz estándar para datos paralelo
(b) una interfaz estándar para datos serie
(c) una mejora de la interfaz IEEE-488
(d) lo mismo que SCSI

11. El bus FireWire es lo mismo que el

(a) bus IEEE 488
(b) USB
(c) IEEE 1394
(d) RS-422
(e) RS-423

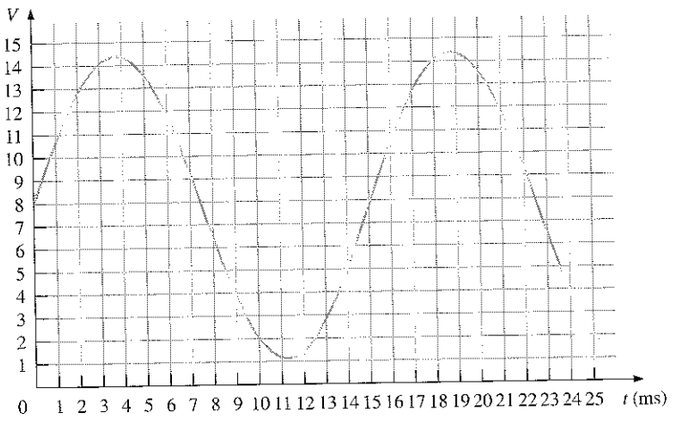
12. El bus USB admite hasta

(a) 63 dispositivos **(b)** 14 dispositivos
(c) 100 dispositivos **(d)** 127 dispositivos

■ PROBLEMAS SECCIÓN 13.1 Interfaces entre el mundo digital y el analógico

1. La curva analógica de la Figura 13.57 se muestrea a intervalos de 1 ms. Representar la curva completa mediante una serie de números binarios de 4 bits.

FIGURA 13.57



2. Dibujar la reproducción digital de la curva del problema 1.

3. Reflejar en una gráfica la función analógica representada por la siguiente secuencia de números binarios:

1111, 1110, 1101, 1100, 1010, 1001, 1000, 0111, 0110, 0101, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1100, 1100, 1011, 1010, 1001.

SECCIÓN 13.2 Conversión digital-analógica (D/A)

4. La tensión de entrada de un circuito amplificador operacional inversor es 10 mV y la salida es de 2 V. ¿Cuál es la ganancia de tensión en bucle cerrado?

5. Para conseguir una ganancia de tensión en bucle cerrado de 330 con un amplificador inversor, ¿qué valor de resistencia de realimentación se debe utilizar si R_i = 1,0 kΩ?

6. En el DAC de 4 bits de la Figura 13.7, la resistencia de menor peso es de 10 kΩ. ¿Cuáles son los valores de las restantes resistencias de entrada?

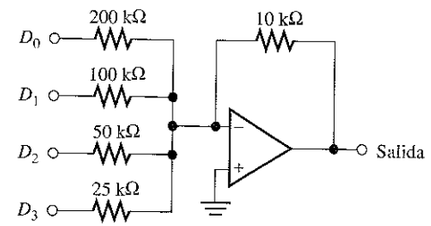
7. Determinar la salida del DAC de la Figura 13.58(a) si se aplica a sus entradas la secuencia de números de 4 bits de la parte (b) de la figura. Los datos de entrada tienen un nivel bajo de 0 V y un nivel alto de +5 V.

8. Repetir el problema 7 para las entradas de la Figura 13.59.

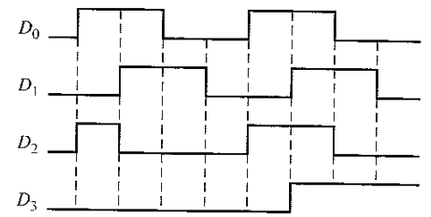
9. Determinar la resolución expresada en porcentaje para cada uno de los siguientes DAC:

(a) 3 bits **(b)** 10 bits **(c)** 18 bits

FIGURA 13.58

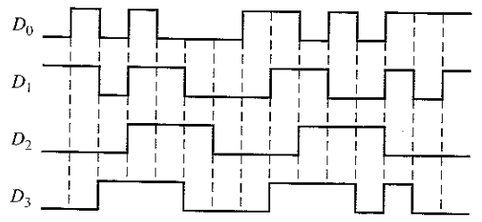


(a)



(b)

FIGURA 13.59



SECCIÓN 13.3 Conversión analógica-digital (A/D)

10. Determinar el código de salida binario de un ADC flash de 3 bits para la señal analógica de entrada de la Figura 13.60. La velocidad de muestreo es 100 kHz.

11. Repetir el problema 10 para la señal analógica de la Figura 13.61.

12. Para un ADC de rampa digital de 3 bits, la tensión de referencia aumenta un escalón cada microsegundo. Determinar la secuencia binaria codificada para la señal analógica de la Figura 13.62.

13. Para un ADC de rampa digital de 4 bits, suponer que el periodo de reloj es 1 μs. Determinar la secuencia binaria de salida para la señal analógica de la Figura 13.63.

FIGURA 13.60

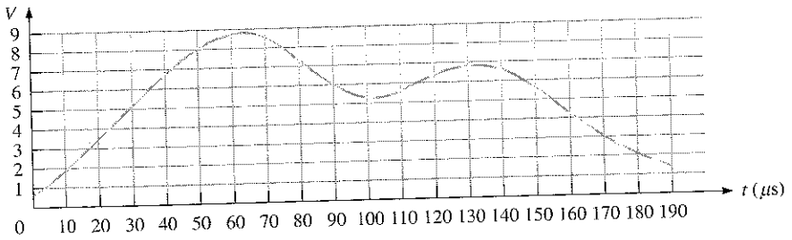
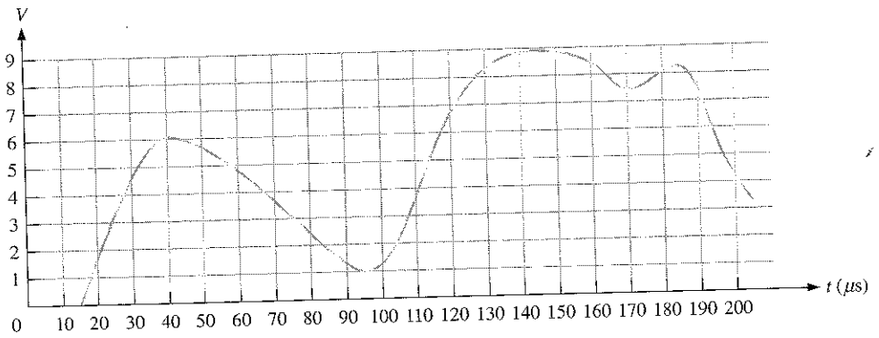


FIGURA 13.61



V

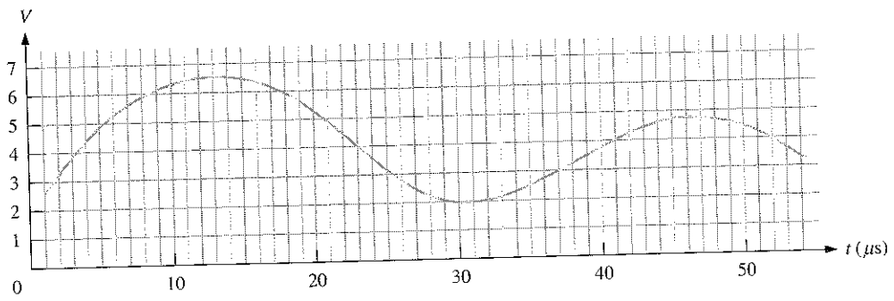


FIGURA 13.62

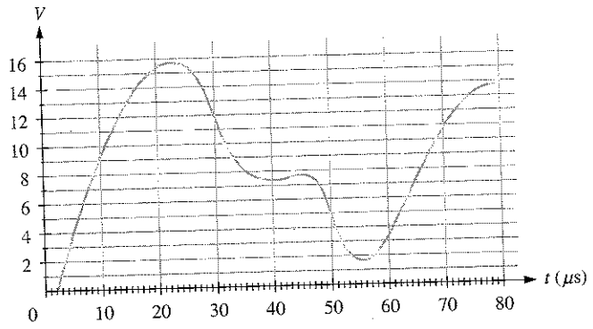


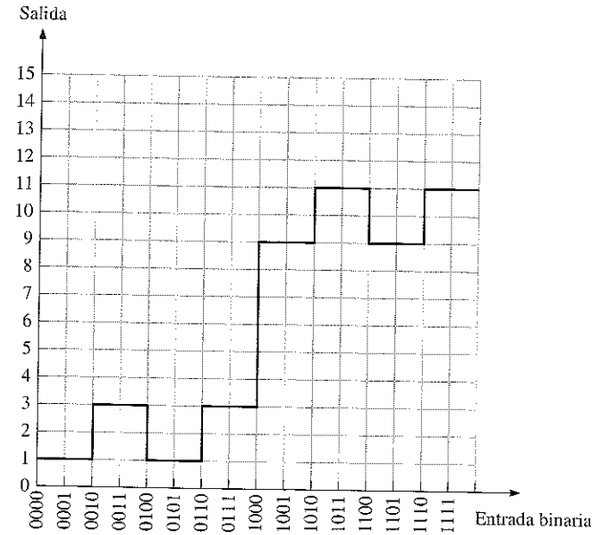
FIGURA 13.63

14. Repetir el problema 12 para un ADC de seguimiento.
15. Para un determinado ADC por aproximaciones sucesivas de 4 bits, el escalón de salida máximo es +8V. Si se aplica una tensión constante de +6 V a la entrada analógica, determinar la secuencia de estados binarios del SAR.

SECCIÓN 13.4 Localización de averías en los convertidores DAC y ADC

16. Desarrollar un circuito que genere una secuencia binaria de prueba de 8 bits para el montaje de prueba de la Figura 13.27.
17. Un DAC de 4 bits falla, de forma que el MSB permanece en el estado 0. Dibujar la salida analógica cuando se aplica a las entradas una secuencia binaria ascendente.

FIGURA 13.64



18. Se aplica una secuencia binaria ascendente a un DAC de 4 bits y se observa la salida indicada en la Figura 13.64. ¿Cuál es el problema?
19. Un ADC genera la siguiente secuencia de números binarios cuando se aplica a sus entradas una señal analógica: 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 0110, 0101, 0100, 0011, 0010, 0001, 0000.
    (a) Reconstruir la entrada digitalmente.
    (b) Si el ADC falla faltando el código 0111, ¿qué aspecto tendría la salida reconstruida?

SECCIÓN 13.5 Interfaces internas del sistema

20. En una transferencia serie de ocho bits de datos desde un dispositivo fuente a un dispositivo aceptor, se observa la secuencia de establecimiento de comunicación de la Figura 13.65 en las cuatro líneas de un bus genérico. Analizando el cronograma, identificar la función de cada señal e indicar si se originan en la fuente o en el aceptor.
21. Determinar la señal en la línea de bus de la Figura 13.66 para las formas de onda de habilitación y de entrada de datos mostradas.
22. En la Figura 13.67(a), dos fuentes colocan datos sobre el bus de datos bajo el control de la línea de selección. La señal de selección se muestra en la Figura 13.67(b). Determinar la forma de onda del bus de datos para los códigos de salida de los dispositivos indicados.

FIGURA 13.65

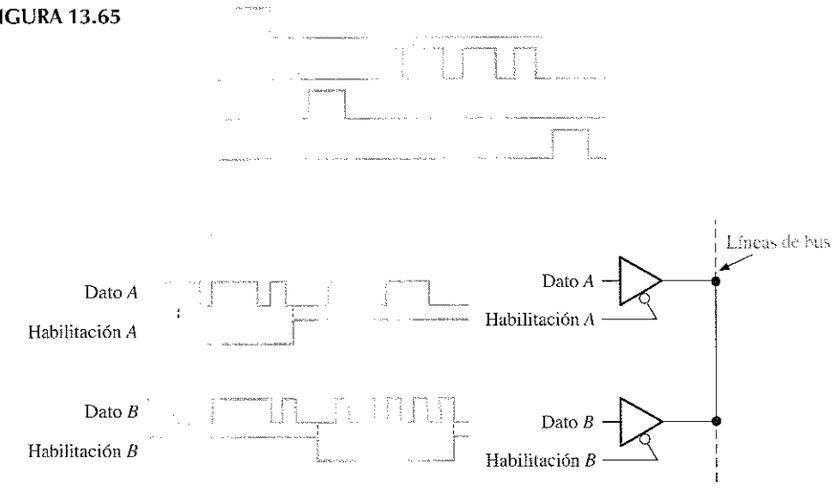


FIGURA 13.66

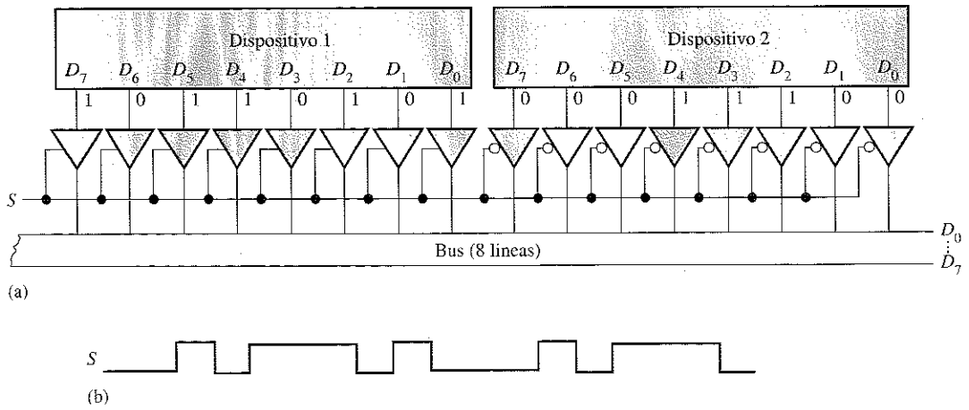


FIGURA 13.67

SECCIÓN 13.6 Buses estándar

23. Explicar la diferencia básica entre un bus local y el bus PCI.

24. Definir "plug-and-play".

25. ¿En qué difieren el bus PCI y el bus ISA?

26. Si se emplea una interfaz RS-232C muy corta, ¿pueden transmitirse los datos a una velocidad muy rápida?

27. ¿De qué especificación de bus son parte los DCE y DTE? Indicar qué significan los acrónimos DCE y DTE.

28. Enumerar los hilos de un cable USB.

29. Ocho instrumentos compatibles GPIB se conectan al bus. ¿Cuántos más pueden añadirse sin exceder las especificaciones?

30. Considerar la interfaz GPIB entre un dispositivo transmisor y otro receptor mostrada en la Figura 13.68(a). A partir del cronograma para el establecimiento de la comunicación de la parte (b), determinar cuántos bytes de datos se transfieren realmente al dispositivo receptor.

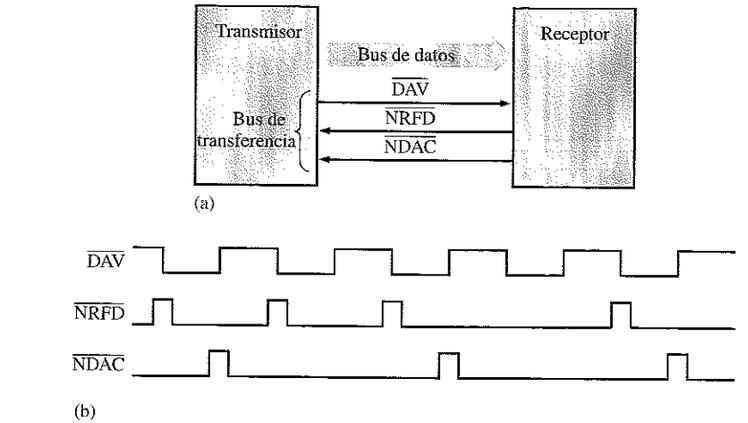


FIGURA 13.68

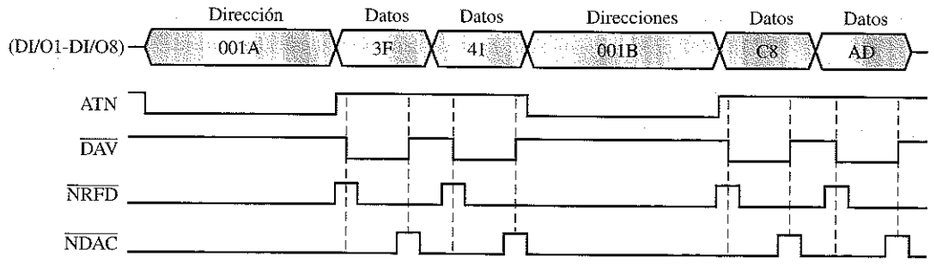


FIGURA 13.69

31. Describir las operaciones mostradas en el cronograma GPIB de la Figura 13.69. Dibujar un diagrama de bloques básico para el sistema implicado en esta operación.

32. Un dispositivo transmisor envía un byte de datos a un dispositivo receptor en un sistema GPIB. Simultáneamente, un DTE envía un byte de datos a un DCE por medio de una interfaz RS-232C. ¿Qué sistema recibirá el primero el byte de datos completo? ¿Por qué?

Aplicación a sistemas

SECCIÓN 13.7 Aplicación a los sistemas digitales

33. Determinar la frecuencia máxima de la señal de entrada de selección, teniendo en cuenta las limitaciones especificadas en la Práctica de sistemas 2.

34. Determinar la resolución del ADC utilizado en el sistema de posicionamiento de antena.

35. Determinar el incremento más pequeño que se puede medir en los ángulos de azimut y elevación suponiendo rangos de 0° a 180°.

36. El sistema de posicionamiento de antena se puede simplificar limitando el posicionamiento a una sola dimensión en cada momento; es decir, primero se fija el azimut y después la elevación. Desarrollar un diagrama de bloques básico para este método.

Éstos requieren el uso del CD-ROM que acompaña al libro.

37. Abrir el archivo PR13_37.EWB y determinar el fallo, si existe, en el circuito.

38. Abrir el archivo PR13_38.EWB y determinar el fallo, si existe, en el circuito.

39. Abrir el archivo PR13_39.EWB y determinar el fallo, si existe, en el circuito.

40. Abrir el archivo PR13_40.EWB y determinar el fallo, si existe, en el circuito.

■ RESPUESTAS A LAS REVISIONES DE CADA SECCIÓN

SECCIÓN 13.1

1. En la naturaleza, las magnitudes son analógicas.
2. Para convertir una magnitud analógica a forma digital se usa la conversión A/D.
3. Para convertir una magnitud digital a forma analógica se usa la conversión D/A.

SECCIÓN 13.2

1. En un DAC con ponderación binaria, cada resistencia tiene un valor diferente.
2. $[1/(2^4 - 1)]\,100 = 6{,}67\%$

SECCIÓN 13.3

1. El método paralelo (flash) es más rápido.
2. El ADC de seguimiento utiliza un contador ascendente/descendente.
3. Sí, el convertidor por aproximaciones sucesivas tiene un tiempo de conversión fijo.

SECCIÓN 13.4

1. Un escalón invertido indica un comportamiento no monotónico en un DAC.
2. Las amplitudes de los escalones en un DAC son menores que la ideal cuando existe una baja ganancia.
3. La existencia de códigos incorrectos o la omisión de códigos son algunos de los posibles errores de salida de un ADC.

SECCIÓN 13.5

1. Los excitadores triestado permiten a los dispositivos desconectarse por completo del bus cuando no están en uso, evitando la interferencia con otros dispositivos.
2. Un bus interconecta todos los dispositivos de un sistema y hace posible la comunicación entre los posibles dispositivos.

SECCIÓN 13.6

1. Transferencia de datos serie y paralelo.
2. **(a)** paralelo **(b)** serie **(c)** serie **(d)** paralelo
3. Serie: un bit al mismo tiempo. Paralelo: 8 o más bits al mismo tiempo.
4. 127 dispositivos USB.
5. El bus FireWire es más rápido que el USB.

SECCIÓN 13.7

1. Se requieren tres dígitos para representar ángulos de 0° a 180°.
2. La posición del cursor del potenciómetro se controla mediante la posición del eje del motor, que corresponde a la posición de la antena. La tensión de salida es proporcional a la posición del cursor.
3. Se requieren dos registros para almacenar los códigos de posición actuales de azimut y elevación.

■ RESPUESTAS A LOS PROBLEMAS RELACIONADOS DE LOS EJEMPLOS

13.1 Véase la Figura 13.70

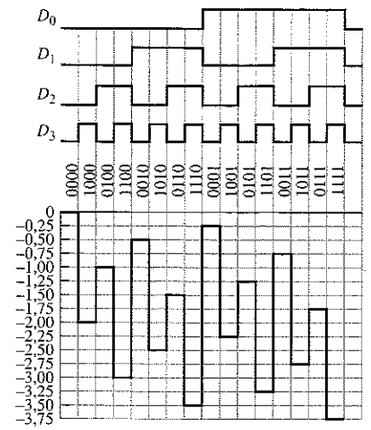


FIGURA 13.70

13.2 $[1/(2^{16} - 1]100 = 0{,}00153\%$

13.3 010, 100, 101, 110, 110, 111, 110, 110, 101, 100, 011, 010, 000, 000, 001, 001, 010, 011, 100, 101, 110, 110, 111, 111

13.4 Véase la Figura 13.71

13.5 Véase la Figura 13.72

FIGURA 13.71

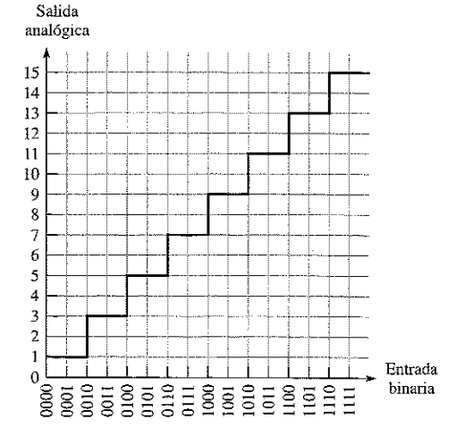
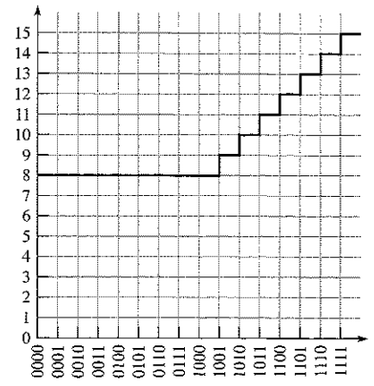


FIGURA 13.72



RESPUESTAS AL AUTOTEST

1. (b) **2.** (c) **3.** (c) **4.** (b) **5.** (a) **6.** (a)
7. (b) **8.** (b) **9.** (b) **10.** (b) **11.** (c) **12.** (d)

14

INTRODUCCIÓN A LOS MICROPROCESADORES Y COMPUTADORAS

14.1 El microprocesador y la computadora
14.2 Familias de microprocesadores
14.3 El microprocesador y el modelo software 8086/8088 para el procesador Pentium
14.4 Programación de microprocesadores
14.5 La unidad central de proceso (CPU)
14.6 La memoria
14.7 El puerto de entrada/salida (E/S)
14.8 Interrupciones
14.9 Acceso directo a memoria (DMA)

WWW Simulación por computadora de circuitos digitales:
El tutorial de Electronics Workbench (EWB) se encuentra en la dirección http://www.prenhall.com/floyd.

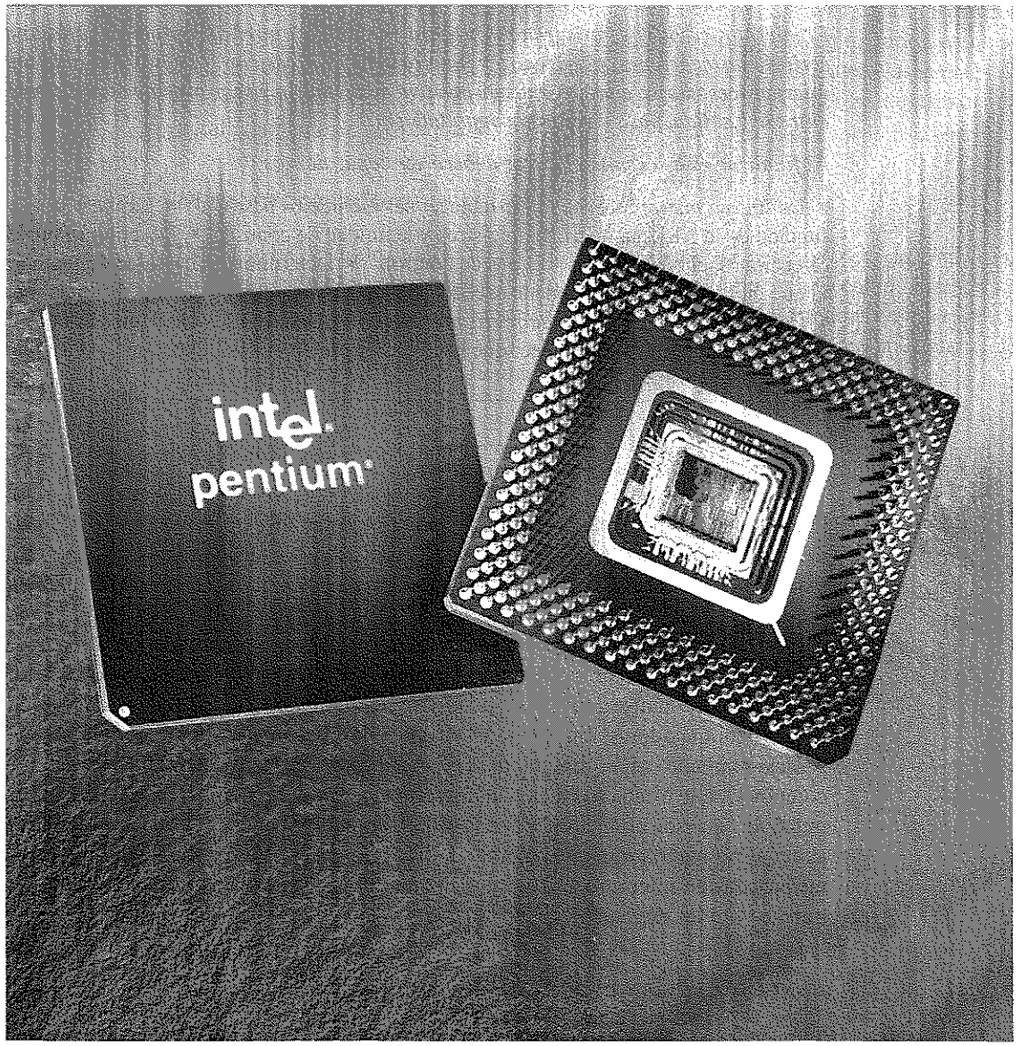
■ OBJETIVOS DEL CAPÍTULO

- ❑ Enumerar los elementos básicos de un microprocesador.
- ❑ Enumerar las unidades básicas de una computadora.
- ❑ Conocer las características y el desarrollo de la familia de microprocesadores Intel.
- ❑ Conocer las características y el desarrollo de la familia de microprocesadores Motorola.
- ❑ Distinguir entre lenguaje ensamblador y lenguaje máquina.
- ❑ Explicar el funcionamiento básico de una CPU 8086/8088.
- ❑ Explicar la arquitectura básica del microprocesador 8086/8088.
- ❑ Explicar el funcionamiento del bus multiplexado del microprocesador 8086/8088.
- ❑ Explicar el modelo software de los procesadores Pentium.
- ❑ Describir un sencillo programa en lenguaje ensamblador.
- ❑ Describir la función del controlador de bus en la CPU 8086/8088.
- ❑ Analizar un cronograma para un ciclo de lectura de memoria y un ciclo de escritura en memoria.
- ❑ Distinguir entre puerto de E/S dedicado y puerto de E/S mapeado en memoria.
- ❑ Comparar la E/S por sondeo, la E/S controlada por interrupción y las interrupciones software.
- ❑ Describir las funciones de los dispositivos PIC y PPI.
- ❑ Definir y explicar la ventaja del acceso directo a memoria (DMA).

■ INTRODUCCIÓN

Este capítulo es una breve introducción a los microprocesadores y computadoras. Naturalmente, lo que se va a cubrir en único capítulo será muy limitado, ya que fácilmente se podrían dedicar uno o más capítulos a cada una de las secciones. Sin embargo, debe tenerse en cuenta que nuestro propósito es proporcionar una introducción básica.

Se estudian brevemente las familias de microprocesadores Intel y Motorola. Se utiliza como modelo el microprocesador 8088 de Intel para ilustrar los conceptos básicos sobre microprocesadores, describiendo en detalle las últimas mejoras. Este método es muy adecuado, ya que el 8086/8088 es la primera generación de la familia de microprocesadores 80X86 de Intel; aunque las nuevas generaciones (el 80286, 80386, 80486 y la familia Pentium) son más potentes y tienen características avanzadas, mantienen una arquitectura y unas funciones básicas comunes, como la estructura de registros.



14.1 ■ EL MICROPROCESADOR Y LA COMPUTADORA

El microprocesador es un circuito integrado digital que puede programarse con una serie de instrucciones, para realizar funciones específicas con los datos. Cuando un microprocesador se conecta a un dispositivo de memoria y se provee de dispositivos de entrada y salida, pasa a ser una computadora.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Definir un *microprocesador* y establecer sus elementos básicos. ■ Explicar la función de la ALU, la matriz de registros y la unidad de control ■ Describir el bus de direcciones, el bus de datos y el bus de control. ■ Determinar la cantidad de memoria a la que un microprocesador puede acceder, en función del tamaño del bus de direcciones. ■ Definir el concepto de *lenguaje ensamblador.* ■ Describir los elementos básicos de una computadora. ■ Establecer qué hace cada parte de una computadora. ■ Explicar qué es un dispositivo periférico.

La revolución de la computadora

Durante los últimos veinticinco años, el microprocesador ha revolucionado el campo de las computadoras. Anteriormente al microprocesador, las computadoras eran extremadamente grandes, muy caras y casi nadie podía poseer una. Todo esto cambió con la introducción del **microprocesador**, un circuito de gran escala de integración, que contenía la unidad central de procesamiento completa de una computadora en un único circuito integrado. Las primeras computadoras que usaron microprocesadores no eran más que computadoras de aficionados y se denominaban "computadoras domésticas" o microcomputadoras. En 1981, IBM entró en el mercado de las computadoras de pequeño tamaño con la computadora personal (normalmente denominada PC, personal computer), que estaba basada en el microprocesador 8088 de Intel. Todavía, a las computadoras que son compatibles con la arquitectura original de Intel se las denomina PC, aunque son mucho más que computadoras "personales". Actualmente, las computadoras basadas en microprocesador se han convertido en instrumentos fundamentales de trabajo para los negocios y la industria, así como para el público general.

Elementos básicos de una computadora

Todas las computadoras constan de bloques funcionales básicos que incluyen una *unidad central de proceso* (CPU), *memoria* y *puertos de entrada/salida.* Estos bloques funcionales se conectan mediante tres buses, como muestra la Figura 14.1. Estos tres buses son el bus de datos, el bus de direcciones y el bus de control. Los dispositivos de entrada y salida se conectan a través de los puertos de entrada y salida. Un **puerto** es una interfaz física de una computadora a través de la cual pasan los datos hacia, y desde, los periféricos.

Las instrucciones y datos se almacenan en memoria en posiciones específicas determinadas por el **programa**, conjunto de instrucciones creado para resolver un problema específico. Cada posición tiene asociada una **dirección** exclusiva. La CPU obtiene las instrucciones colocando una dirección en el bus de direcciones. Las instrucciones se transfieren a través del bus de datos cuando son solicitadas por la CPU.

La CPU ejecuta las instrucciones secuencialmente; con frecuencia, las instrucciones modifican los datos almacenados en memoria u obtenidas a través de un dispositivo de entrada. Los datos procesados pueden almacenarse de nuevo en memoria o enviarse a un dispositivo de salida a través del bus de datos. La CPU genera las señales del bus de control para coordinar todas estas operaciones, las cuales se describen más detalladamente en este capítulo

Microprocesadores

Como se ha definido anteriormente, un microprocesador es un circuito de gran escala de integración que contiene la CPU completa de una computadora en un único circuito integrado. Según han ido evolucionando las computadoras basadas en microprocesador, éstas han comenzado a ser cada vez más potentes y han ido reemplazando a las computadoras mainframe tradicionales en los negocios y en la industria.

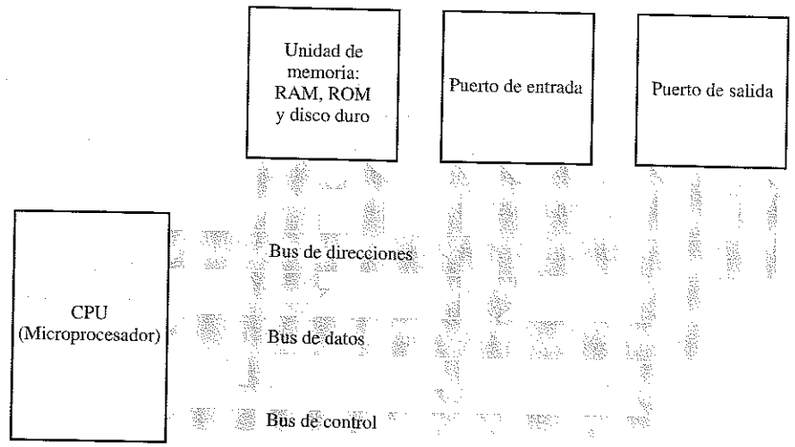


FIGURA 14.1
Diagrama de bloques básico de una computadora.

Los microprocesadores se usan en las estaciones de trabajo y en los servidores, así como en dispositivos periféricos como son impresoras y las unidades de disco. Los microprocesadores se emplean también en muchas aplicaciones que requieren funciones de control y supervisión, entre las que se incluyen el control industrial y de maquinaria, los controles de motores y la telemetría, por nombrar algunas. En muchas de estas aplicaciones de control, un primo del microprocesador, el **microcontrolador**, ha encontrado un muy amplio campo de aplicación.

Un microprocesador contiene varias unidades, cada una de ellas diseñada para realizar un trabajo específico. Las unidades en sí, su diseño y organización, se denominan **arquitectura** del microprocesador. La arquitectura determina el conjunto de instrucciones y el proceso de ejecución de dichas instrucciones. En la Figura 14.2 se muestran tres de la unidades básicas principales, comunes a todos los microprocesa-

dores: la *unidad aritmético lógica* (ALU), la *matriz de registros* y una *unidad de control*. Existen otras unidades que trabajan junto con estas tres unidades básicas para formar un microprocesador específico.

FIGURA 14.2
Los tres elementos básicos de un microprocesador.

Una interesante forma de microprocesador lo constituye el **coprocesador**. Un coprocesador es realmente un microprocesador diseñado con un conjunto limitado de instrucciones optimizado para realizar operaciones aritméticas muy rápidamente. A medida que los microprocesadores han evolucionado, uno de los cambios ha consistido en la incorporación del coprocesador dentro de lo que originalmente era sólo el microprocesador. El coprocesador para el Pentium está incluido dentro del propio microprocesador Pentium.

Unidad aritmético lógica. La ALU es el elemento clave de procesamiento del microprocesador. Está gobernada por la unidad de control para realizar operaciones aritméticas como la suma y la sustracción, así como operaciones lógicas como NOT, AND, OR y OR-exclusiva. Los datos para la ALU se obtienen de la matriz de registros.

Matriz de registros. La **matriz de registros** es la colección de registros contenidos en el microprocesador. Durante la ejecución de un programa, los datos y direcciones se almacenan temporalmente en los registros que conforman esta matriz. La ALU puede acceder a los registros muy rápidamente, haciendo que el programa se ejecuten de forma eficaz. Algunos registros se clasifican como registros de propósito general, lo que significa que se pueden emplear para cualquier propósito dictado por el programa. Otros registros tienen capacidades y funciones específicas y no se pueden usar como registros de propósito general. Existen otros registros adicionales que se denominan *registros invisibles para el programa*, utilizados únicamente por el procesador y que no están disponibles para el programador.

Unidad de control. La **unidad de control** se encarga de procesar las instrucciones. Proporciona las señales de temporización y control para extraer e introducir los datos en el microprocesador y para sincronizar la ejecución de las instrucciones.

Buses del microprocesador

Los tres buses que hemos mencionado anteriormente son parte de las conexiones internas y externas que utilizan los microprocesadores para desplazar datos, direcciones y señales de control.

El bus de direcciones. El **bus de direcciones** es una "calle de un solo sentido", a través de la cual el microprocesador envía un código de dirección a una memoria o a un dispositivo externo. El tamaño o anchura del bus de direcciones se especifica mediante el número de conductores. Los primeros microprocesadores utilizados en computadoras tenían 16 líneas de dirección, con las que se podía direccionar 65.536 (2^{16}) posiciones distintas, es decir 64k. Cuantos más bits tenga el bus de direcciones, a más posiciones de memoria puede acceder un microprocesador. El número de líneas de dirección se ha ido incrementando a 20, 24 y 32 bits a medida que avanzaba la tecnología de los microprocesadores. El Pentium, con 32 líneas de dirección, permite acceder a, aproximadamente, 4.295.000.000 (4G) posiciones de memoria.

El bus de datos. El **bus de datos** es una "calle de dos sentidos", por la que los datos o instrucciones se transfieren al microprocesador, o por la que los resultados de una operación o cálculo son enviados por el microprocesador. Los primeros microprocesadores tenían buses de datos de 8 bits. Dependiendo de cada microprocesador concreto, el tamaño del bus de datos es de 8, 16, 32 o 64 bits.

El bus de control. El microprocesador utiliza el **bus de control** para coordinar las operaciones y comunicarse con los dispositivos externos. El bus de control tiene señales que permiten leer y escribir datos en la memoria o en un puerto de entrada/salida en el instante apropiado. Las líneas de control también se utilizan para insertar estados de espera para los dispositivos más lentos y evitar la contienda de bus, una condición que puede producirse cuando dos o más dispositivos intentan transmitir al mismo tiempo.

Programación de microprocesadores

Todos los microprocesadores trabajan con un conjunto de instrucciones básico, formulado por los diseñadores del procesador. Por ejemplo, el Pentium dispone de cientos de variaciones de sus instrucciones, que se clasifican en siete grupos básicos:

- Transferencia de datos
- Aritmética
- Manipulación de bits
- Bucles y saltos
- Cadenas de caracteres
- Subrutinas e interrupciones
- Control del procesador

Cada instrucción consta de una cadena binaria (1s y 0s) que el microprocesador decodifica antes de ser ejecutada. En la Sección 14.4 se describen estos grupos de instrucciones. Las instrucciones en código binario se denominan *lenguaje máquina* y son las únicas que reconoce el microprocesador. Las primeras computadoras se programaban escribiendo realmente las instrucciones en código binario, lo que era un trabajo tedioso y propenso a errores.

Para simplificar la tarea de escribir las instrucciones para una computadora, se creó el lenguaje ensamblador. El **lenguaje ensamblador** se clasifica como un lenguaje de bajo nivel, ya que las instrucciones que emplea se pueden traducir directamente a instrucciones en código binario o patrones de datos. Como ilustra la Figura 14.3, un **programa ensamblador** convierte las instrucciones (escritas con una sinta-

xis similar al inglés y denominadas **mnemónicos**) del lenguaje ensamblador en lenguaje máquina (patrones binarios), que es el utilizado por el microprocesador. Otras instrucciones utilizadas en lenguaje ensamblador describen estructuras de datos y otras informaciones que necesita el programa ensamblador. Estas instrucciones se denominan **pseudo-operaciones**, ya que no se traducen a instrucciones reales que emplee el microprocesador.

FIGURA 14.3
Diagrama de bloques para la programación del microprocesador.

El lenguaje ensamblador y el correspondiente lenguaje máquina son específicos para el tipo de microprocesador o familia de microprocesadores. Por ejemplo, las instrucciones escritas en lenguaje ensamblador para los procesadores de la serie 600 de Motorola no funcionan en la familia de procesadores de Intel. La ventaja del lenguaje ensamblador es que el programador tiene un control directo del proceso y, normalmente, los programas escritos en dicho lenguaje se ejecutan más rápidamente. Además, el programador puede gestionar mejor los procesos de la máquina, como son las interrupciones o los dispositivos de E/S, con el lenguaje ensamblador que con la mayoría de los restantes lenguajes de computadora.

Por otra parte, los lenguajes de programación de alto nivel como BASIC, Pascal, C o FORTRAN son independientes del tipo de microprocesador que tenga un sistema informático. Un programa llamado **compilador** o intérprete traduce las instrucciones de un programa escrito en un lenguaje de alto nivel a lenguaje máquina. La ventaja es que se puede diseñar un lenguaje de propósito general para problemas específicos, tales como las aplicaciones de negocios. Para la resolución de problemas de carácter general, los lenguajes de alto nivel son mejores que el lenguaje ensamblador. En la Sección 14.4 se expone más detalladamente el tema de la programación.

Un sistema de computadora

Para que una computadora realice una tarea determinada, debe comunicarse con el "mundo exterior" estableciendo una interfaz con el usuario, con los dispositivos de detección o con los dispositivos que haya que controlar. En la Figura 14.4 se muestra un sistema de computadora típico. Normalmente, se emplea un teclado para la introducción de datos, un monitor de vídeo y una impresora. En la mayoría de las computadoras también se utiliza un ratón. Frecuentemente, otros tipos de periféricos (como unidades de disco externas, escáneres, tarjetas gráficas o dispositivos de entrada de voz) también forman parte del sistema.

Unidad central de proceso. La parte de la computadora que controla la ejecución de un programa es la **CPU** que, en las computadoras basadas en microprocesador, es el propio microprocesador. Básicamente, la CPU direcciona una posición de memoria, obtiene (carga) la instrucción de programa que está almacenada en dicha posición y ejecuta la instrucción. Después de completar una instrucción, la CPU pasa a la siguiente (los más recientes microprocesadores pueden trabajar simultáneamente en más de una instrucción). Este proceso de carga y ejecución se repite hasta que se hayan ejecutado todas las instrucciones de un programa específico. Un ejemplo sencillo de un programa de aplicación sería un conjunto de instrucciones almacenado en memoria en forma de códigos binarios, que haga que la CPU cargue una serie de números también almacenados en la memoria, los sume y almacene la suma de nuevo en memoria.

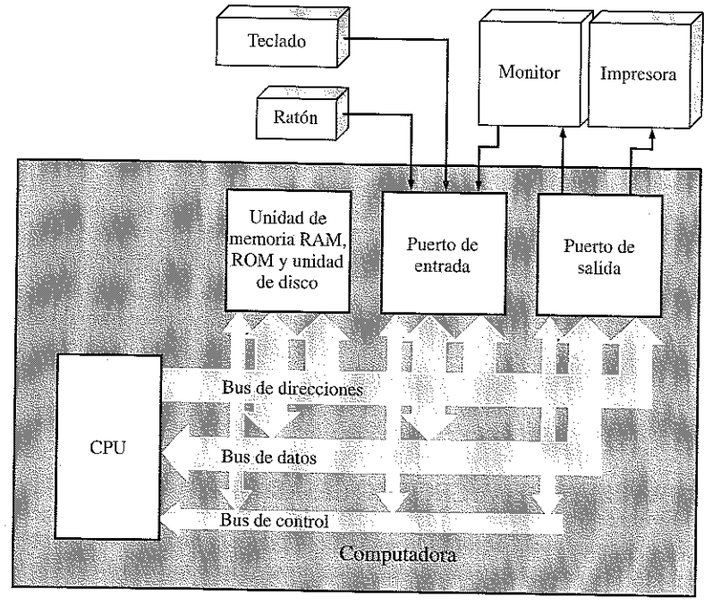


FIGURA 14.4
Sistema de computadora típico.

Unidad de memoria. Normalmente, la unidad de memoria consta de una RAM, una ROM y memorias de disco duro. La RAM almacena temporalmente datos y programas durante el procesamiento. Los datos son números y otras informaciones en formato binario y los programas son listas de instrucciones. Dado que, generalmente, la RAM es una memoria volátil, toda información debe copiarse en un dispositivo de almacenamiento en disco no volátil.

La ROM almacena los programas del sistema, como por ejemplo el **BIOS** (Basic Input/Output System). Este programa contiene determinadas instrucciones que indican a la CPU lo que tiene que hacer cuando se enciende la computadora por primera vez y también dirige el autotest de inicialización, informando de cualquier error que se produzca. También contiene rutinas para controlar el monitor de vídeo, los gráficos, el teclado, las comunicaciones asíncronas, así como otros periféricos. Otros programas pueden acceder a estas rutinas a través de una serie de interrupciones de software.

Puertos de entrada. Generalmente, la computadora recibe información de un dispositivo de entrada a través de un puerto de entrada. Algunos microprocesadores no utilizan puertos, sino que se comunican con los dispositivos de E/S a través de posiciones de memoria específicamente reservadas para el dispositivo de E/S, como veremos en la Sección 14.7. El teclado, el ratón y el disco duro son ejemplos de dispositivos de entrada a los que accede la CPU, normalmente, a través de puertos de entrada individuales.

Puertos de salida. La computadora envía información a los dispositivos de salida a través de un puerto de salida (o a través de posiciones de memoria reservadas). Un ejemplo de un periférico con el que una computadora se comunica a través de un puerto de salida es el monitor de vídeo. Algunos periféricos funcionan como dispositivos de entrada y de salida, como son las unidades de disco externas y los modems. En los dispositivos que requieren comunicación en los dos sentidos, se configuran un puerto de salida y de entrada distintos.

REVISIÓN DE LA SECCIÓN 14.1

1. Nombrar los elementos básicos de una computadora.
2. Describir cada uno de los tres buses utilizados por un microprocesador.
3. ¿Cuáles son los dos tipos de información que se almacenan en memoria?
4. ¿Qué es el lenguaje ensamblador? ¿Qué dos tipos de operaciones existen en un lenguaje ensamblador?

14.2 ■ FAMILIAS DE MICROPROCESADORES

Intel Corporation es una compañía pionera en el desarrollo y comercialización de microprocesadores. Actualmente, tiene una cuota del mercado mundial de los microprocesadores muy grande, ya que sus procesadores se utilizan en la mayoría de las computadoras compatibles PC. Esta sección expone brevemente la historia y describe la evolución de la familia de microprocesadores 80X86, desde el 8086/8088 hasta el Pentium I, II y III. Motorola también ha disfrutado de un considerable éxito con su línea de microprocesadores. La primera computadora Apple estaba basada en el 6800 de Motorola, y las computadoras Macintosh de Apple actuales utilizan microprocesadores Motorola. Al igual que la familia 80X86 de Intel, los microprocesadores Motorola se han construido a partir del diseño básico establecido por el primer dispositivo, añadiéndole características y mejoras de funcionamiento. Nos centraremos en las características principales mejoras introducidas de una generación a la siguiente. Téngase en mente que la tecnología de los microprocesadores avanza rápidamente y que los parámetros de funcionamiento cambian a medida que los dispositivos actuales se mejoran o son reemplazados por las nuevas versiones.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Exponer los aspectos fundamentales de la familia de microprocesadores 80X86 de Intel. ■ Exponer los aspectos fundamentales de la familia de microprocesadores 680X0 de Motorola. ■ Describir una memoria caché y establecer su finalidad ■ Describir el concepto de arquitectura pipeline y establecer su propósito

Los primeros microprocesadores Intel

El primer microprocesador de Intel fue el 4004, introducido en 1971. Fue diseñado para la calculadora Busicom y tenía un bus de datos de sólo 4 bits. A este dispositivo le siguió el 8008, que tenía un bus de datos de 8 bits y fue utilizado por Don Lancaster, un aficionado a la informática, para crear una computadora. A mediados de los años 70, se introdujeron otros dos microprocesadores de 8 bits, el 8080 y el 8085. Estos dos dispositivos podían direccionar sólo 2^{16} = 65.536 posiciones de memoria (64 kbytes de memoria) con sus buses de direcciones de 16 bits. El microprocesador 8080 fue empleado en la primera computadora puesta a disposición del público, la computadora Altair.

La familia de microprocesadores 80X86

Desde su introducción en 1978, la llamada arquitectura 80X86 ha experimentado numerosas fases de evolución. La primera generación de la familia 80X86 incluye el 8086, el 8088 y el 80186. Después surgieron el 80286, seguido del 80386 y luego el 80486. El Pentium es la quinta generación de microprocesadores Intel, a la que ahora siguen los microprocesadores Pentium Pro, Pentium II y Pentium III, a los que normalmente se les denomina simplemente procesadores. Otra versión del microprocesador Pentium, denominada Celeron, fue introducida en 1998. Cada nuevo microprocesador se ha diseñado partiendo de los conceptos básicos del 8086/8088, añadiendo características adicionales y mejoras de rendimiento.

Existen tres métodos básicos para mejorar la capacidad de un microprocesador para procesar datos. Los diseñadores de chips siempre buscan aumentar la velocidad de la señal de reloj, aunque esto da lugar a un problema importante, que consiste en que el circuito disipa más potencia (aumenta el calor) cuando la velocidad de la señal de reloj se incrementa. Un segundo método consiste en incrementar el tamaño del bus de datos, para llevar más información al procesador. Por último, la arquitectura interna se puede cambiar para ejecutar instrucciones de forma más eficiente. Un importante cambio en la arquitectura interna fue la introducción del procesamiento y la duplicación de las unidades de ejecución, para procesamiento en paralelo. A medida que repase la historia de los microprocesadores, comprobará que cada uno de los tres métodos mencionados ha sido fundamental para la mejora del rendimiento de los nuevos microprocesadores y son partes clave de las especificaciones de los mismos.

El 8086/8088 y 80186. Desarrollado en 1978, el 8086 fue el primero de la familia 80X86 y es la base de todos los microprocesadores Intel que le siguieron. El 8086 era

un microprocesador de 16 bits (bus de datos de 16 bits) y representó un avance significativo respecto a los dispositivos anteriores de 8 bits. El número de líneas de dirección del 8086 se aumentó a veinte, pudiendo direccionar 2^{20} =1.048.576 posiciones de memoria (1 Mbyte). El 8086 representaba una mejora de un orden de magnitud con respecto a su predecesor, el procesador 8080. Las distintas versiones del 8086 funcionaban a frecuencias de reloj de 5 MHz, 8 MHz y 10 MHz.

El 8088 es, esencialmente, un 8086 con un bus de datos interno de 16 bits multiplexado como bus de datos externo de 8 bits. Fue desarrollado para cubrir la demanda de aplicaciones en los sistemas, más simples, de 8 bits y se utilizó en la computadora IBM PC original. El éxito de esta computadora llevó a que la revista Fortune nombrara a la compañía IBM como una de las empresas triunfadoras de los años setenta.

El 80186 es un 8086 con varias funciones auxiliares integradas en el chip, tales como generador de reloj, controlador del sistema, controlador de interrupciones y controlador de acceso directo a memoria (DMA, direct memory access). Se aumentó la frecuencia de reloj a 12,5 MHz y la velocidad de 5 MHz disponible en el 8086/8088 se eliminó, quedando las frecuencias de 8 MHz, 10 MHz y 12,5 MHz. El 80186 representaba un avance relativamente pequeño en cuanto al funcionamiento respecto del 8086, por lo que nunca fue empleado como CPU en computadoras comerciales.

El 80286 (286). En 1982 apareció el 80286, cuya capacidad de direccionamiento de memoria era ya de 24 líneas de dirección, lo que permitía acceder a 2^{24} = 16.777.216 posiciones de memoria (16 Mbytes). Pero más importante fue el modo de operación avanzado denominado *modo protegido,* que se incorporó por primera vez en el 80286, y luego ha sido utilizado por todos los microprocesadores posteriores. Este modo permite el acceso a posiciones de memoria adicionales y a características de programación avanzada, como la **multitarea**, un entorno de sistema operativo en el que la computadora parece ejecutar múltiples programas o tareas simultáneamente. Además, el 286 añadió una nueva unidad funcional de direccionamiento dedicada al direccionamiento virtual, un concepto que permite que un programa largo se ejecute en una memoria física más pequeña. Aunque el 80286 operaba con las mismas frecuencias de reloj que el 80186, representó un importante cambio en la arquitectura y añadió nuevas instrucciones dedicadas a las operaciones multitarea. Y, a pesar de todas estas mejoras, el 286 podía ejecutar todo el código escrito para el antiguo 8086.

El 80386 (386). En 1985, se dio un gran paso cuando Intel presentó el microprocesador de 32 bits. El 80386 tiene un bus de datos de 32 bits y un bus de direcciones de 32 bits. La versión DX tiene un bus de datos externo de 32 bits, mientras que la versión SX tiene un bus de datos externo de 16 bits. Con 32 bits de dirección se puede acceder a 2^{32} = 4.294.967.296 posiciones de memoria (4Gbytes).

Al 80386 se le añadió un nuevo mecanismo de paginación que permitía un uso más eficaz de la memoria, manteniendo la necesaria compatibilidad con los procesadores anteriores. El proceso de formación de la dirección física se repartió en dos niveles de tablas de búsqueda, en las que se almacenaba toda la información acerca de las direcciones. Este proceso permitía que cualquier programa escrito para el 8086 tuviera su propia área de memoria de 1 Mbyte, como si estuviera ejecutándose él solo en la memoria original del 8086. Los sistemas operativos (como OS/2) podían ejecutar varias tareas de forma rotatoria, de modo que varios usuarios pudieran acceder de forma simultánea a los programas.

El 386 es compatible con todos los modelos anteriores, incluyendo la matriz de registros. Aunque los registros de propósito general son de 16 bits en todas las máquinas más antiguas, en el 386 se ampliaron a 32 bits proporcionando importantes nuevas capacidades. Además de otras funciones, el 80386 fue el primer microprocesador Intel que aumentó la velocidad de proceso con el uso de una pipeline de instrucciones. Esta técnica permite al microprocesador comenzar a trabajar en una nueva instrucción antes de haber completado la instrucción en curso. Existen versiones del 80386 que funcionan a 16 MHz, 20 MHz, 25 MHz y 33 MHz.

Existen otras versiones más económicas del 80386, como son el 80386SX y el 80386SL, en las que el bus de datos se multiplexa a 16 bits y el bus de direcciones se multiplexa a 24 bits. Todos los microprocesadores 80386 pueden funcionar con un coprocesador matemático, que puede utilizarse para mejorar o colaborar con las funciones del microprocesador, mediante la realización de operaciones en coma flotante.

El 80486 (486). El siguiente paso en el proceso de evolución fue la introducción, en 1989, del 80486, con una memoria caché interna de 8 kbytes. Como hemos visto en el Capítulo 12, la memoria caché reduce el tiempo de acceso a memoria, almacenando las instrucciones o los datos recién utilizados en una SRAM, más rápida que la relativamente lenta memoria principal (DRAM). También se incorporó un coprocesador matemático interno y la arquitectura interna se rediseñó para acelerar determinadas instrucciones. Aunque los cambios en el modelo de programación entre el 386 y el 486 fueron relativamente pocos, la incorporación de la memoria caché y otras mejoras dieron lugar a un procesador significativamente mejor. El 80486 fue el primer microprocesador de Intel en contener más de un millón de transistores. Ciertas versiones del 80486 pueden funcionar a frecuencias de reloj superiores a 66 MHz.

El Pentium. El Pentium, que apareció en 1993, mantiene el bus de direcciones de 32 bits del 80486, pero duplica el bus de datos a 64 bits. El Pentium mantiene los registros de 32 bits del 486, aunque permite utilizar el bus de datos de 64 bits por el procedimiento de añadir una segunda unidad de ejecución. El Pentium también dispone de dos memorias caché de 8 kbytes, una para las instrucciones y otra para los datos. Un método de doble pipeline, conocido como *arquitectura superescalar*, aumenta la velocidad a la que se pueden procesar las instrucciones, superando la pipelinización única introducida en el 80386. El Pentium incorpora circuitería interna para reconocer cuándo pueden emplearse ambas pipelines; en estos casos, el Pentium puede ejecutar dos instrucciones a la vez. Otro importante avance fue la introducción de un nuevo coprocesador matemático muy mejorado. El Pentium original operaba a frecuencias de reloj de 60 MHz o 66 MHz.

Pentium Pro. El Pentium Pro, denominado P6, apareció en 1995 y empleaba la arquitectura básica del Pentium original. Diseñado para computadoras de alta gama, estaciones de trabajo y servidores, el Pentium Pro contenía tres unidades para operaciones con enteros, otra unidad para coma flotante y memoria caché de niveles uno y dos. El Pentium Pro se suministra con un segundo chip de memoria caché, que permite mejorar la velocidad. Opera a frecuencias de reloj de hasta 200 MHz.

Pentium II. El Pentium II apareció en 1997 e incorpora la tecnología MMX de Intel, diseñada específicamente para procesar vídeo, sonido y datos gráficos de forma

eficiente. La tecnología MMX mejora la compresión/descompresión de vídeo, la manipulación de imágenes, el cifrado y el procesamiento de E/S. El Pentium II dispone de 57 nuevas instrucciones, diseñadas para acelerar las secuencias repetitivas que a menudo se encuentran en las operaciones multimedia. El Pentium II es compatible con todos los procesadores anteriores, desde el 8086. En 1998, aparecieron miembros adicionales de la familia Pentium II, entre los que se incluye el procesador Celeron, diseñado para el mercado del PC básico. En 1999, la frecuencia de reloj del Pentium II era de hasta 450 MHz y la del procesador Celeron alcanzaba los 400 MHz. En el futuro se esperan más mejoras en cuanto a velocidad.

Pentium III. El Pentium III fue introducido en 1999. Fue diseñado para el mercado de las computadoras de escritorio destinadas al consumidor general y a las empresas. El Pentium III incluye 70 nuevas instrucciones, diseñadas para mejorar las aplicaciones 3D, de imágenes y de vídeo. Las primeras computadoras que incorporaron este procesador operaban a 500 MHz, aunque el Pentium III ha sido probado, por Intel, a velocidades superiores a 1 GHz, usando técnicas especiales de enfriamiento.

El 6800 de Motorola y sus variaciones

Los primeros microprocesadores de Motorola eran todos dispositivos de 8 bits (bus de datos de 8 bits). El 6800 apareció en 1975 con una frecuencia de reloj de 2 MHz y era capaz de direccionar 64 kbytes de memoria con un bus de direcciones de 16 bits. La arquitectura interna incluía dos acumuladores, un contador de programa, un registro de índice y un puntero de pila, pero ningún registro de propósito general. Un **acumulador** es un registro con capacidades especiales para realizar operaciones aritméticas y lógicas. Para el almacenamiento de datos, el 6800 utiliza memoria RAM en lugar de registros de propósito general.

En el 6802 se añadió una RAM de 128 bytes para utilizarla en lugar de determinados registros. La frecuencia de reloj se incrementó en el 6803 a 3,58 MHz y se añadió una UART (universal asynchronous receiver/transmitter, transmisor/receptor asíncrono universal) para las comunicaciones serie. El último microprocesador de 8 bits fue el 6809, que ofrecía un juego de instrucciones mayor, incluyendo la instrucción de multiplicación. Todos los dispositivos de 8 bits disponían de un bus de direcciones de 16 bits.

Los microprocesadores de la familia 680X0

El 68000. El 68000 fue introducido en 1979 y fue el primero de los microprocesadores de 16 bits de Motorola. Aunque está clasificado como un procesador de 16 bits, internamente tenía registros de 32 bits. También disponía de 24 líneas de dirección, con las que se podía acceder a 16 Mbytes de memoria (por comparación con el 8086/8088, con una capacidad de 1 Mbyte). Con una frecuencia de reloj de 16 MHz, el 68000 era más rápido que los dispositivos Intel de la misma época. Asimismo, el concepto de registro basado en RAM utilizado en el 6800 fue abandonado en favor de una matriz de registros interna, que incluía ocho registros de propósito general de 32 bits, siete registros de direcciones de 32 bits, dos punteros de pila de 32 bits, así como un contador de programa de 32 bits y un registro de estado de 16 bits. Los procesadores que le siguieron eran todos compatibles en sentido ascendente, es decir, los programas escritos para el 68000 pueden ejecutarse en todos los microprocesadores de la familia 680X0.

El 68020. En 1985, Motorola entró oficialmente en el mundo de los microprocesadores de 32 bits con el 68020, que tenía un mejor rendimiento respecto de los procesadores de 16 bits. Se podían direccionar 4 Gbytes de memoria (empleando un bus de direcciones de 32 bits) y disponía de una memoria caché de 256 bytes para las instrucciones, una funcionalidad innovadora en dicha época. La frecuencia de reloj se aumentó a 33 MHz y estaban disponibles un coprocesador matemático (el 68882) y otros chips auxiliares.

El 68030. El 68030 fue introducido en 1991; este dispositivo añadía otra memoria caché de 256 bytes para los datos, y la velocidad de operación se incrementó a 50 MHz. El bus de direcciones se amplió a 32 bits. Con éstas y otras importantes mejoras que redundaban en una mayor eficiencia, el 68030 podía ejecutar instrucciones a una velocidad más de dos veces superior a la de su predecesor inmediato.

El 68040. Introducido en 1995, el 68040 aumentó las memorias caché para instrucciones y datos a 4 kbytes, y se incorporó al chip un coprocesador matemático para la realización de operaciones en coma flotante. El procesador también disponía de múltiples pipelines independientes para permitir la ejecución simultánea de instrucciones, así como de una unidad interna para operaciones en coma flotante y un mecanismo integrado de gestión de memoria.

El 68060. El 68060 se introdujo también en 1995 y se caracterizaba por una arquitectura superescalar, con una pipeline múltiple de instrucciones y dos memorias caché de 8 kbytes (para datos e instrucciones). El consumo de potencia se redujo disminuyendo la tensión de operación a 3,3 V. La frecuencia máxima de reloj es de 75 MHz.

El PowerPC. El MPC601 apareció en 1992 como el primero de una serie de microprocesadores de 64 bits con arquitectura superescalar, que permite ejecutar de forma efectiva hasta tres instrucciones por ciclo de reloj. Dispone de 32 kbytes de memoria caché y de coprocesador matemático interno. El PowerPC es un microprocesador tipo RISC. **RISC** significa Reduced Instruction Set Computer (computadora con conjunto de instrucciones reducido), un método que reduce el número de instrucciones disponibles, pero optimiza el reducido conjunto de instrucciones y da lugar a un mejor rendimiento global. El último microprocesador incorporado a la familia Power PC es el 750, anunciado para 1999. Dispone de una memoria caché de 32 kbytes para instrucciones y de otra para datos también de 32 kbytes. La velocidad del 750 es de 400 MHz.

REVISIÓN DE LA SECCIÓN 14.2

1. ¿Cuál es el significado del término *microprocesador de 32 bits*?
2. ¿Cuál fue el primer microprocesador de 16 bits de Intel?
3. ¿Qué quiere decir RISC?
4. ¿Cuál es la finalidad de la memoria caché?
5. ¿Qué es una arquitectura superescalar?

14.3 ■ EL MICROPROCESADOR Y EL MODELO SOFTWARE 8086/8088 PARA EL PROCESADOR PENTIUM

Como hemos dicho en la sección anterior, la familia de microprocesadores 80X86 ha experimentado un cambio tremendo desde el 8086/8088 hasta la familia Pentium, tanto en velocidad como en complejidad. Sin embargo, el conjunto básico de registros y otras funcionalidades del 8086/8088 se han mantenido (y ampliado) a lo largo de su proceso evolutivo, de manera que todos los nuevos procesadores de Intel respondan a las mismas instrucciones (así como a una serie de nuevas instrucciones) que el original 8086/8088. Esta sección comienza con una breve introducción al 8086/8088, para presentar los conceptos básicos sobre la arquitectura, operación y programación del microprocesador. La sección termina con una corta introducción a los principales cambios que se han producido en la estructura de registros que define el modelo software de los más recientes procesadores. El método que vamos a seguir es mostrar un procesador básico (el 8086/8088) y comentar las mejoras introducidas en la línea Intel a lo largo de su evolución.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar el funcionamiento básico del microprocesador. ■ Describir la unidad de interfaz con el bus. ■ Indicar el propósito de los registros de segmento. ■ Explicar el propósito del puntero de instrucción. ■ Describir la unidad de ejecución. ■ Describir el conjunto general de registros. ■ Explicar el propósito del registro de indicadores (flags).

Funcionamiento básico

Un microprocesador ejecuta un programa (lista de instrucciones) repitiendo cíclicamente los tres pasos siguientes:

1. Extrae una instrucción de la memoria y la coloca en la CPU.
2. Decodifica la instrucción; si la instrucción requiere alguna otra información, extrae dicha información. En este paso, el contador de programa se actualiza para apuntar a la siguiente instrucción.
3. Ejecuta la instrucción (hace lo que dice la instrucción). Devuelve los resultados a los registros y a la memoria.

Anteriormente al microprocesador 8086/8088, estos pasos se realizaban de forma secuencial; el procesador 8085 no podía extraer una instrucción al mismo tiempo que estaba ejecutando otra. La arquitectura del 8086/8088 era fundamentalmente diferente. Disponía de dos unidades internas distintas, la unidad de ejecución (**EU**; Execution Unit), que ejecuta la instrucción, y la unidad de interfaz del bus (**BIU**, Bus Interface Unit), que actúa de interfaz con los buses del sistema y extrae las instrucciones, lee los operandos y escribe los resultados. En la Figura 14.5 se muestran estas unidades.

La BIU lleva a cabo todas las operaciones con el bus para la unidad de ejecución, tal como la transferencia de datos desde la memoria o la E/S. Mientras que la EU ejecuta las instrucciones, la BIU "busca" y extrae más instrucciones de la memoria. Esta acción se denomina **precarga** o *pipelinización*. El concepto de precarga permite que el procesador ejecute instrucciones al mismo tiempo que se extrae la instrucción

siguiente, eliminando tiempo de inactividad. Las instrucciones precargadas se almacenan en una memoria interna de alta velocidad llamada **cola** de ejecución. Esta cola permite a la BIU mantener alimentada a la unidad de ejecución con instrucciones, de modo que ésta no tenga que esperar a que se cargue la siguiente instrucción de la memoria, sino que se recupera directamente de la cola en mucho menos tiempo. En el Pentium, este proceso va un paso más allá. Hay disponibles dos unidades de ejecución completas que permiten la ejecución de dos instrucciones al mismo tiempo de forma independiente. Determinados compiladores se diseñan para aprovechar esta ventaja de disponer de dos unidades de ejecución, mediante un proceso conocido como **emparejamiento de instrucciones**, que elimina las dependencias.

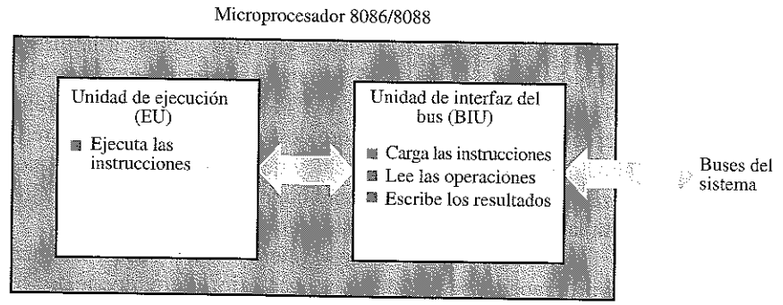


FIGURA 14.5
Las dos unidades internas del 8086/8088: unidad de ejecución (EU) y unidad de interfaz con el bus (BIU).

Arquitectura básica del 8086/8088

La Figura 14.6 presenta un diagrama de bloques de la arquitectura (organización interna) de un microprocesador 8088. Externamente, el 8088 tiene 20 bits de dirección, que permiten acceder a 1 Mbyte (1.048.576 bytes) de memoria, y utiliza un bus de datos de 8 bits. Internamente, el 8088 tiene un bus de datos de 16 bits y una cola de 4 bytes. El 8086 es idéntico, excepto en que tiene un bus de datos externo de 16 bits y una cola de instrucciones de 6 bytes.

La unidad de interfaz con el bus (BIU)

Las partes principales de la BIU son: la cola de instrucciones de 4 bytes, los registros de segmento (CS, DS, SS y ES), el puntero de instrucción (IP) y el bloque sumador de direcciones (Σ). Los buses de datos internos de 16 bits y el bus de cola Q interno interconectan la BIU y la unidad de ejecución (EU).

Cola de instrucciones. La cola de instrucciones aumenta la velocidad media a la que el programa se ejecuta (llamada **tasa de procesamiento**), almacenando hasta cuatro bytes (seis en el 8086). Como se ha mencionado anteriormente, esta técnica permite que el 8088 haga esencialmente dos cosas a la vez: carga y ejecución. Esta función se ha ampliado en los procesadores posteriores, para incluir colas más rápidas y de mayor tamaño.

Registros de segmento. Los procesadores 8086/8088 tienen cuatro registros de segmento (CS, DS, SS y ES), que son registros de 16 bits utilizados en el proceso de formación de direcciones de 20 bits. Un **segmento** es un bloque de memoria de 64 kbytes que puede comenzar en cualquier posición del espacio de memoria de 1 Mbyte (1.048.576 bytes), supuesto que comienza en una frontera de 16 bytes, es decir, que es divisible por 16.

En el proceso de diseño del 8086/8088 y de los procesadores siguientes, Intel seleccionó un método original de generación de direcciones físicas de 20 bits, utilizando dos registros de 16 bits. Uno de los registros que forman la dirección física (real) es siempre un registro de segmento; el otro registro es un registro de propósito general de 16 bits, que contiene información de dirección. La principal ventaja de este método era que permitía reubicar código fácilmente. Un **código reubicable** puede moverse a cualquier lugar dentro del espacio de memoria, sin tener que cambiar el código básico.

Cada uno de los cuatro segmentos identifica la dirección de comienzo de un bloque de 64 kbytes (65.536 bytes), que representa una "ventana" en el espacio de memoria (20 bits) completo de 1 Mbyte. La dirección de comienzo de un segmento se representa mediante el número de 16 bits contenido en el registro de segmento, más 4 bits que se añaden a la derecha y que se supone siempre que son cero. En otras palabras, los registros de segmento contienen los 16 bits más significativos que representan la dirección física de comienzo del segmento.

Los cuatro registros de segmento (CS, DS, SS y ES) pueden ser modificados por el programa para apuntar, si es necesario, a otro bloque de 64 kbytes. Para fragmentos de código cortos no es necesario cambiar los segmentos. Los cuatro segmentos pueden encontrarse en ubicaciones separadas dentro del espacio de memoria o pueden solaparse, dependiendo del tamaño de los requisitos del código en concreto. Pueden incluso definirse como el mismo bloque de 64 kbytes. En el 8086/8088, los segmentos de memoria direccionables en cada momento determinado son aquellos que quedan definidos por las direcciones de segmento contenidas en los registros CS (code segment, segmento de código), DS (data segment, segmento de datos), SS (stack segment, segmento de pila) y ES (extra segment, segmento extra o adicional). En los procesadores posteriores, se añadieron otros registros de segmento.

Como se ha mencionado, cada segmento ocupa 64 kbytes de memoria. Para encontrar una posición de memoria determinada, se combina una dirección de segmento con un desplazamiento. La dirección del segmento representa los dieciséis bits más significativos (cuatro dígitos en hexadecimal) de la dirección física donde comienza un segmento. El desplazamiento son dieciséis bits adicionales que representan la distancia desde el inicio del segmento hasta una dirección física situada dentro del segmento. La Figura 14.7 ilustra cómo se divide la memoria en segmentos y desplazamientos y muestra ejemplos de segmentos solapados y no solapados.

Puntero de instrucción (IP) y bloque sumador de direcciones. El puntero de instrucción (**IP**, instruction pointer) de 16 bits indica el desplazamiento correspondiente a la siguiente instrucción en memoria que se va a ejecutar. El IP siempre hace referencia al registro CS (segmento de código); por tanto, la dirección física de la siguiente instrucción se forma combinando el segmento de código y el puntero de instrucción. El puntero de instrucción siempre contiene el desplazamiento de la siguiente instrucción, y el registro CS siempre contiene la dirección del segmento. En lenguaje ensamblador, esta dirección se expresa como CS:IP.

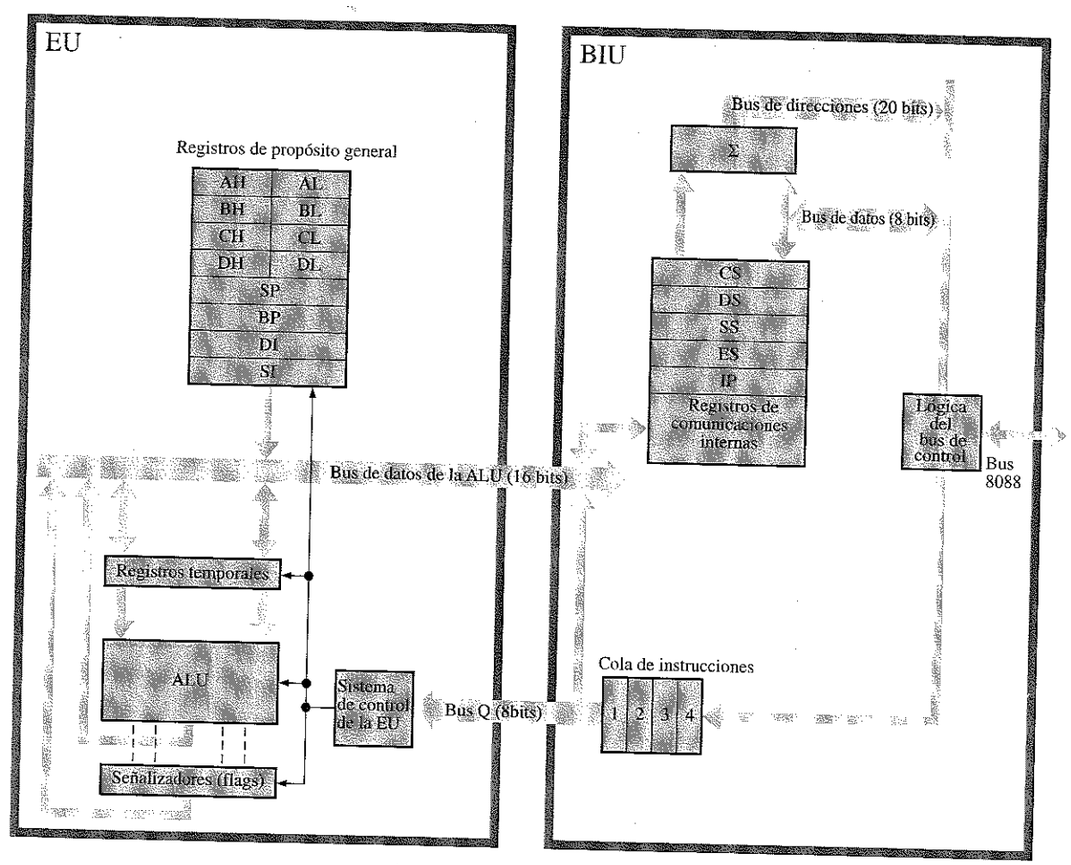


FIGURA 14.6
Organización interna del microprocesador 8088.

Para formar la dirección física de 20 bits de la siguiente instrucción, se añade el desplazamiento de 16 bits contenido en IP a la dirección del segmento almacenada en el registro CS, la cual se desplaza cuatro bits a la izquierda, como se indica en la Figura 14.8. Como se ha mencionado anteriormente, se supone que la posición menos significativa contiene el valor binario 0000. Esta suma la realiza el bloque sumador de instrucciones.

La Figura 14.9 ilustra el direccionamiento de una posición de memoria por el método segmento:desplazamiento. En la figura, $A000_{16}$ es el contenido del registro de segmento y $A0B0_{16}$ el de IP. Cuando el segmento de registro CS se desplaza y se suma al contenido de IP se obtiene $A0000_{16} + A0B0_{16} = AA0B0_{16}$, que es la dirección física.

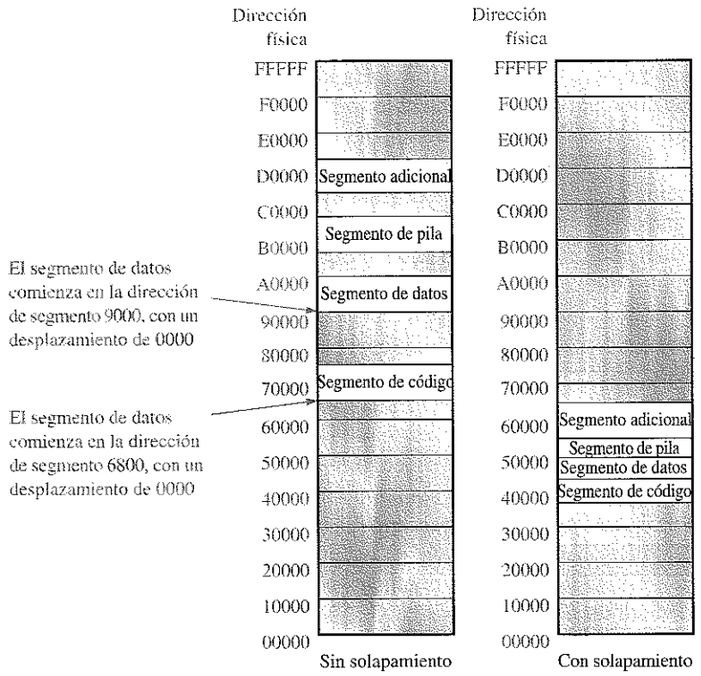


FIGURA 14.7
Segmentos solapados y no solapados en el primer Mbyte de memoria. Cada segmento representa 64 kbytes.

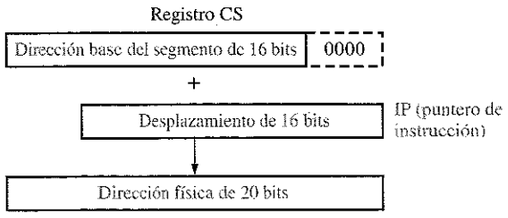


FIGURA 14.8
Formación de la dirección física de 20 bits a partir de la dirección base del segmento y el desplazamiento.

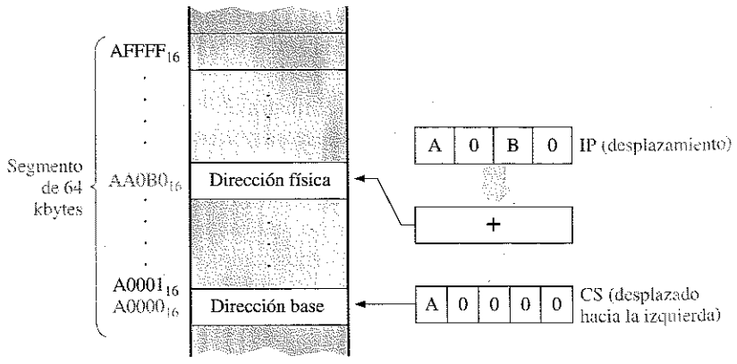
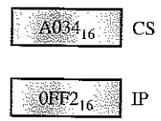


FIGURA 14.9
Ilustración del método de direccionamiento por segmentos.

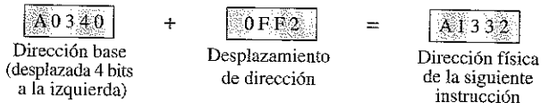
EJEMPLO 14.1 En la Figura 14.11 se muestra el contenido de los registros CS e IP. Determinar la dirección física de memoria de la siguiente instrucción.

FIGURA 14.10



Solución. La dirección base contenida en CS se desplaza 4 bits a la izquierda (un dígito hexadecimal) y se coloca un 0_{16} en la posición menos significativa, como se muestra en la Figura 14.11. Se suman la dirección base desplazada y el desplazamiento para obtener la dirección física de 20 bits.

FIGURA 14.11



Problema relacionado. Determinar la dirección física si el registro CS contiene el número hexadecimal $6B4D_{16}$.

Es importante entender cómo se usa el método segmento:desplazamiento para formar la dirección física; sin embargo, en programación, no es normalmente necesario que el programador haga esto para especificar la dirección física real. Este trabajo es realizado por el programa ensamblador, utilizando etiquetas suministradas por el programador. Cuando se requiere una dirección física, el programador la especifica generalmente empleando el método segmento:desplazamiento. Por tanto, la dirección del Ejemplo 14.1 podría proporcionarse simplemente como A034:0FF2.

La unidad de ejecución

La unidad de ejecución (EU) decodifica las instrucciones cargadas por la BIU, genera las señales de control apropiadas y ejecuta las instrucciones. Las partes fundamentales de la EU son la unidad aritmético-lógica (ALU), los registros de propósito general y los indicadores.

La unidad aritmético-lógica (ALU). Esta unidad realiza todas las operaciones lógicas y aritméticas. Trabaja con operandos de 8 bits o 16 bits.

Los registros de propósito general. Este conjunto de registros de 16 bits se divide en dos grupos de cuatro registros cada uno, como se muestra en la Figura 14.12. Uno de los grupos constituye los registros de datos y el otro contiene los registros de índice y de puntero. Generalmente, los registros de **puntero** e índice se usan para almacenar desplazamientos (en este contexto, un puntero hace referencia a una posi-

ción de memoria). En el caso del puntero de pila (SP) y el puntero de base (BP), la referencia predeterminada para formar una dirección física es el segmento de pila (SS). Los punteros de índice (SI y DI) y el registro de base (BX), generalmente adoptan como referencia predeterminada el registro de segmento de datos (DS); para determinadas instrucciones, se hace una excepción a esta regla general.

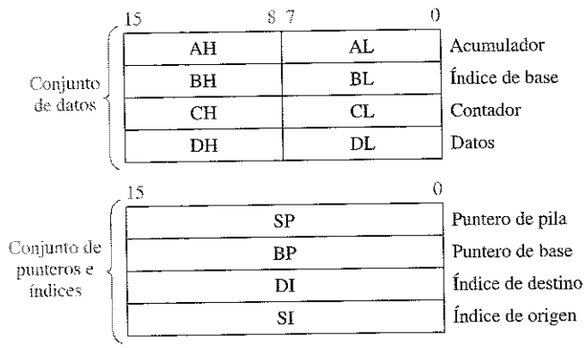


FIGURA 14.12
Conjunto de registros de propósito general del 8086/8088.

Cada uno de los registros de 16 bits (AX, BX, CX, DX) tiene dos secciones de 8 bits, accesibles por separado. Dependiendo del programa, se pueden utilizar como registros de 16 bits o como dos registros de 8 bits. Los bytes de menor orden de los registros de datos se denominan *AL, BL, CL* y *DL*. Los bytes de mayor orden se denominan *AH, BH, CH* y *DH*. Estos registros se pueden utilizar en la mayoría de las operaciones aritméticas y lógicas de cualquier forma que especifique el programador, para almacenar los datos antes y después del procesamiento. Determinadas instrucciones de programa utilizan específicamente algunos de estos registros.

Los registros de puntero e índice son el puntero de pila (SP, stack pointer), el puntero de base (BP, base pointer), el índice de origen (SI, source index) y el índice de destino (DI, destination index). Estos registros se utilizan para el direccionamiento de la memoria bajo el control de la EU.

Los indicadores (flags). Como muestra la Figura 14.13, el registro de indicadores contiene nueve bits de estado y de control independientes. Un indicador es un bit de estado, utilizado para reflejar una determinada condición después de que la ALU haya realizado una operación lógica o aritmética, tal como un acarreo (CF), un resultado cero (ZF) o el signo de un resultado (SF), entre otros. Los indicadores de control se usan para alterar las operaciones del procesador en determinadas situaciones.

Modelo software de la familia de procesadores Pentium

A medida que Intel ha ido introduciendo sus nuevos microprocesadores, sus funcionalidades y velocidad han aumentado enormemente. Con el procesador Pentium, el concepto de pipeline introducido en el 8086/8088 se amplió para utilizar dos pipelines para

procesamiento de números enteros. El coprocesador externo se incorporó al microprocesador y los buses de datos y direcciones se ampliaron considerablemente. Otras mejoras (como la frecuencia de la señal de reloj, la reducción en el número de ciclos de reloj requeridos por las instrucciones, la capacidad de predicción de saltos y una unidad de coma flotante) han hecho del Pentium un procesador significativamente mejor que sus predecesores. Además de las mejoras del procesador, se han producido otras muchas mejoras en otras partes de las computadoras (como son los protocolos y el tamaño de bus, la velocidad, el tamaño de memoria y el coste). A pesar de todos estos cambios, los diseñadores de los nuevos procesadores consiguieron mantener la compatibilidad ascendente; es decir, el más nuevo de los procesadores Pentium puede ejecutar el software destinado a cualquiera de sus antecesores. Esto se consigue manteniendo el modelo software básico (estructura de registros) del 8086/8088 original.

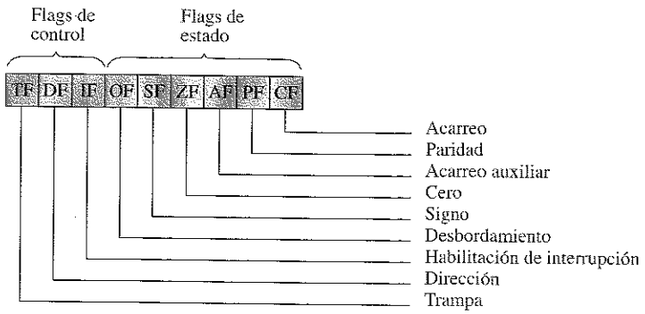


FIGURA 14.13
Indicadores de estado y de control del 8088.

Los registros descritos anteriormente para el 8086/8088 son un subconjunto de los registros de la familia de procesadores Pentium. Con el procesador 80386, el conjunto de registros se amplió para incluir registros de 32 bits. Los registros de 32 bits conservan sus nombres originales, aunque se les añadió la letra E (de Extended) como prefijo; por tanto, la designación de 32 bits para el registro AX es EAX. Además, se añadieron dos nuevos registros de segmento. En la Figura 14.14 se muestran los registros extendidos. Las áreas sombreadas representan los registros que sólo están disponibles en el 386 y procesadores posteriores.

Con la aparición de los nuevos procesadores, además de introducir los registros extendidos, se aumentó el espacio direccionable de memoria de forma importante. Para mantener la compatibilidad ascendente, Intel reserva el primer Mbyte de memoria para los programas que se ejecutan en modo real. El **modo real** es cualquier operación que sólo permite que el procesador acceda al primer Mbyte de memoria, con el fin de simular el comportamiento del procesador 8086/8088. El código escrito para un procesador anterior se puede ejecutar en modo real en cualquiera de los procesadores más recientes (aunque la inversa no es siempre cierta). Generalmente, el código escrito en modo real es compatible (con algunas excepciones) con todos los procesadores Intel, desde el 8086/8088 hacia arriba.

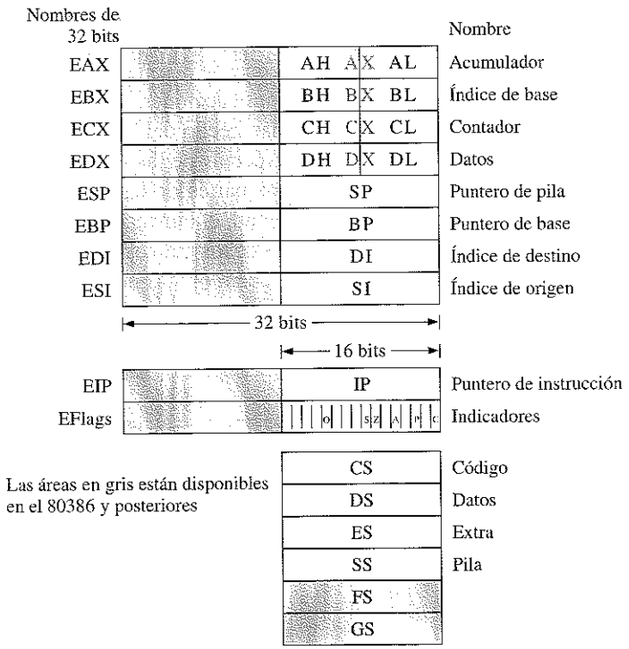


FIGURA 14.14
Registros de los procesadores Intel, desde el 8086/8088 al Pentium.

REVISIÓN DE LA SECCIÓN 14.3

1. Nombrar los registros de propósito general del microprocesador 8086/8088.
2. ¿Cuál es el propósito de la unidad de interfaz con el bus (BIU)?
3. ¿Está conectada la unidad de ejecución con el sistema de buses?
4. ¿Cuál es la función de la cola de instrucciones?
5. ¿Cuál es la ventaja del método segmento:desplazamiento para formar direcciones?
6. ¿Qué es el emparejamiento de instrucciones?

14.4 ■ PROGRAMACIÓN DE MICROPROCESADORES

Todas las computadoras deben programarse para realizar la más elemental de las tareas. Esta sección se centra en el concepto básico de programación en lenguaje ensamblador, un lenguaje de bajo nivel que consiste en una representación simbólica del lenguaje máquina básico utilizado por los microprocesadores.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar el propósito general de un programa de computadora. ■ Describir un sencillo programa en lenguaje ensamblador. ■ Explicar cómo se ejecuta un programa en un microprocesador. ■ Describir los siete grupos de instrucciones de los procesadores 80X86 y Pentium.

La circuitería y componentes físicos (**hardware**) de una computadora son inútiles sin un programa (**software**). Como se ha dicho anteriormente, un programa es una lista de instrucciones organizadas para conseguir un resultado específico. Para dar una idea básica de cómo una computadora lleva a cabo una tarea bajo control de programa, vamos a ver dos sencillos programas en lenguaje ensamblador y los emplearemos para ilustrar el proceso que se sigue en cualquier computadora. Una descripción completa sobre el lenguaje ensamblador queda fuera del ámbito de este libro.

Los programas de esta sección proporcionarán una idea acerca de cómo opera el software en una computadora. Aunque sólo se van a presentar unas pocas instrucciones, la familia de microprocesadores 80X86 dispone de una larga lista de instrucciones, que constituyen el fundamento de una computadora CISC (Complex Instruction Set Computer, computadora con conjunto de instrucciones complejo). Estos procesadores, con su complejo conjunto de instrucciones, ofrecen una gran flexibilidad y potencia de programación. Debido a la amplia base instalada de software que depende de este conjunto de instrucciones, es muy improbable que dicho conjunto se cambie significativamente en un futuro.

En los años ochenta, surgió otra filosofía de diseño que se concentraba en un conjunto reducido de instrucciones, pero que optimizaba su ejecución. Los procesadores que emplean esta arquitectura se denominaron computadoras RISC (Reduced Instruction Set Computer) y han constituido la base de computadoras como la línea Power PC de Apple. Para mantener la exposición dentro de unos límites razonables, vamos a concentrarnos únicamente en los procesadores Intel basados en el 8086/8088 original.

Al final, cualquier programa o conjunto de instrucciones debe proporcionarse al microprocesador en lenguaje máquina o **código máquina**, una serie de códigos binarios que el procesador sea capaz de entender. Aunque sea posible programar en lenguaje máquina, el proceso es extremadamente complejo, propenso a errores y consume mucho tiempo. El lenguaje ensamblador es el lenguaje de nivel inmediatamente superior al código máquina. En lenguaje ensamblador, las instrucciones en código máquina se reemplazan por un mnemónico, un código abreviado de caracteres alfabéticos que es sencillo de recordar. Frecuentemente, a los mnemónicos se les denomina **códigos de operación**. Normalmente, el código de operación va acompañado por uno o dos operandos. Un **operando** es una variable, un registro, una posición de memoria o un valor. La traducción de mnemónicos y operandos escritos por el programador a código máquina que el microprocesador comprenda se consigue mediante el programa ensamblador, explicado anteriormente. Existe una correspondencia uno a uno entre cada instrucción en lenguaje ensamblador y el código máquina generado por el programa ensamblador.

Para ver un ejemplo del proceso de programación, vamos a suponer que queremos que la computadora sume una lista de números sin signo que se encuentran almacenados en memoria, y que coloque la suma de nuevo en memoria. El último número de la lista es cero y se emplea como un indicador de fin de datos. Los pasos que hay que seguir en este ejemplo son los siguientes:

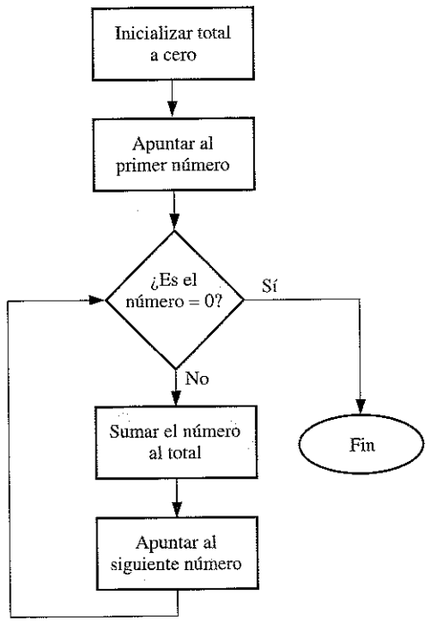
1. Borrar el registro que almacenará el total.
2. Apuntar al primer número que se encuentra en memoria.
3. Comprobar si el número es cero. Si lo es, fin del problema.
4. Si el número es distinto de cero, añadir el número almacenado en memoria al total.

NOTAS INFORMÁTICAS
Grace Hopper, una matemática y pionera de la programación, adquirió una experiencia considerable en la localización de fallos cuando era oficial de la Armada y trabajaba con la computadora Harvard Mark I en los años cuarenta. Ella localizó y documentó en el registro de Mark I, el primer error real de programación. Este error se produjo porque una polilla se quedó atrapada en uno de los relés electromecánicos internos de la máquina, haciendo que la computadora funcionara mal. A partir de entonces, cuando se les preguntaba si habían conseguido algo, los que trabajaban en la computadora respondían que estaban desinsectando (debugging) el sistema. El término debug (que en castellano se traduce como depurar) cuajó, y hoy en día se utiliza para referirse a la localización de problemas en una computadora (u otro dispositivo electrónico) y, especialmente, en el *software*.

5. Apuntar al siguiente número.
6. Repetir los pasos 3, 4 y 5.

Estos pasos se reflejan en el organigrama de la Figura 14.15.

FIGURA 14.15
Organigrama para sumar una lista de números sin signo.



El siguiente programa en lenguaje ensamblador implementa el problema de la suma representado por el organigrama de la Figura 14.15. Obsérvese que los corchetes se utilizan para indicar que un registro está "apuntando" a una posición de memoria. Este programa contiene también una directiva de ensamblador (¿puede encontrarla?). Una **directiva de ensamblador** (también denomina pseudo-operación) es una instrucción dirigida al programa ensamblador, no al microprocesador. En el listado del programa se muestran dos etiquetas: "next" y "done". Estas etiquetas representan una dirección. Los comentarios (no usados por el procesador) van precedidos por un punto y coma.

```
    mov ax, 0               ; borrar el registro ax para almacenar el total
    mov bx, 50H             ; apuntar a una posición de memoria (50H)
                            ; donde comiencen los datos
next: cmp word ptr [bx], 0  ; comprobar si el número al que se apunta es cero
    jz done                 ; si es cero, ir a "done", ya que éste es el
                            ; último dato
    add ax, [bx]            ; añadir la posición de memoria a la que apunta bx
                            ; al total almacenado en ax
    add bx, 02              ; apuntar al siguiente número (dos bytes
                            ; por palabra)
    jmp next                ; repetir el proceso
```



```
done: mov [bx], ax          ; reemplazar el cero almacenado en memoria con el
                            ; total almacenado en ax
    nop                     ; no realiza ninguna operación; se incluye aquí
                            ; para indicar el final.
```

Dependiendo del programa ensamblador, la mayoría de los programas en lenguaje ensamblador contendrán una serie de directivas que son empleadas por el programa ensamblador para llevar a cabo diversas tareas. Estas tareas incluyen la configuración de segmentos, el uso del apropiado juego de instrucciones, la descripción de los tamaños de los datos y la realización de muchas otras funciones de carácter básico. Para simplificar la explicación, en el programa anterior sólo se ha incluido una única directiva, que es imprescindible. La directiva era **word ptr**, que se emplea para indicar el tamaño de los datos a los que apunta el registro BX.

El ensamblador Debug

Si lo desea, haciendo unos pocos cambios, es posible ejecutar el programa anterior empleando un programa ensamblador incorporado, que está disponible en los PC basados en el sistema operativo DOS. Con él, se podrá observar cómo se ejecuta paso a paso. Todos los PC basados en DOS disponen de un programa denominado **Debug**, que incluye un programa ensamblador rudimentario. Para utilizarlo, acceda al sistema DOS y escriba **Debug <cr>** en la línea de comandos (**<cr>** corresponde a "carriage return", retorno de carro, que es lo mismo que presionar la tecla Intro). Debería visualizarse un signo menos, que corresponde al indicativo de Debug. Esta utilidad dispone de una serie de comandos que permiten ver o introducir datos o programas. Antes de escribir y ejecutar un programa en ensamblador, pruebe a introducir algún dato, escribiendo la información que se muestra, en negrita, a continuación:

```
-a 50
```

Esto indica a Debug que comience a ensamblar instrucciones en el segmento de datos actual con un desplazamiento de 50H. Aunque vamos a introducir datos, y no instrucciones, es más sencillo introducir palabras de 16 bits usando este método. Téngase en cuenta que, en Debug, todos los datos se especifican en hexadecimal. Debug responde con una dirección de segmento, que sin duda será diferente de la que se muestra en el listado siguiente (20D8), pero esto no es importante. El desplazamiento (50H) será el mismo. Escribir la información resaltada en negrita (recuérdese que cada **<cr>** quiere decir que hay que pulsar la tecla Intro).

```
20D8:0050 dw 30 <cr>
20D8:0052 dw 15 <cr>
20D8:0054 dw a0 <cr>
20D8:0056 dw 0c <cr>
20D8:0058 dw 00 <cr>
<cr>
```

La instrucción **dw** es una directiva de ensamblador. No se almacena, sino que simplemente informa al programa ensamblador de que cada dato tiene dos bytes de longitud. En los datos que se muestran, sólo se utiliza un byte, pero el programa guardará cada dato en dos posiciones, incluyendo un cero en la posición de mayor orden.

Ahora vamos a introducir el programa en la posición 100; para ello, utilizaremos las instrucciones siguientes:

```
          -a 100 <cr>
20D8:0100 mov ax,0 <cr>
20D8:0103 mov bx,50 <cr>
20D8:0106 cmp word ptr [bx],0 <cr>
20D8:0109 jz 112 <cr>
20D8:010B add ax,[bx] <cr>
20D8:010D add bx,2 <cr>
20D8:0110 jmp 106 <cr>
20D8:0112 mov [bx],ax <cr>
20D8:0114 nop <cr>
20D8:0115 <cr>
```

Para confirmar que se ha introducido correctamente el programa, se puede escribir **u 100 114** en el indicativo de Debug, y el código que haya escrito se mostrará en pantalla. Se visualizará en letras mayúsculas. Ahora escriba **r** a continuación del indicativo de Debug y se mostrará una lista de registros de 16 bits y del estado de los indicadores. Obsérvese que el registro IP debería contener 100, que es la dirección de comienzo del código. Debajo de la lista de registros, se mostrará la primera instrucción (MOV AX, 0000).

Puede utilizarse Debug para ejecutar esta instrucción con el comando **t** (trace), lo que presentará en pantalla el último estado de todos los registros y mostrará la siguiente instrucción (MOV BX, 0050). Ejecutar esta instrucción empleando un comando **t** mostrará que el número 0050 ha sido desplazado al registro BX. En la Figura 14.16 se muestran los pasos hasta este punto de la ejecución. Obsérvese que el primer dato se muestra en la esquina inferior derecha.

```
-u 100 114
20D8:0100 B80000          MOV     AX,0000
20D8:0103 BB5000          MOV     BX,0050
20D8:0106 833F00          CMP     WORD PTR [BX],+00
20D8:0109 7407            JZ      0112
20D8:010B 0307            ADD     AX,[BX]
20D8:010D 83C302          ADD     BX,+02
20D8:0110 EBF4            JMP     0106
20D8:0112 8907            MOV     [BX],AX
20D8:0114 90              NOP
-r
AX=0000   BX=0000   CX=0000   DX=0000   SP=FFEE   BP=0000   SI=0000   DI=0000
DS=20D8   ES=20D8   SS=20D8   CS=20D8   IP=0100     NV UP EI PL ZR NA PE NC
20D8:0100 B8000           MOV     AX,0000
-t

AX=0000   BX=0000   CX=0000   DX=0000   SP=FFEE   BP=0000   SI=0000   DI=0000
DS=20D8   ES=20D8   SS=20D8   CS=20D8   IP=0103     NV UP EI PL ZR NA PE NC
20D8:0103 BB5000          MOV     BX,0050
-t

AX=0000   BX=0050   CX=0000   DX=0000   SP=FFEE   BP=0000   SI=0000   DI=0000
DS=20D8   ES=20D8   SS=20D8   CS=20D8   IP=0106     NV UP EI PL ZR NA PE NC
20D8:0106 833F00          CMP     WORD PTR [BX],+00                   DS:0050=0030
-
```

FIGURA 14.16
Pasos iniciales de ejecución del programa de suma con Debug.


```
DS=20D8   ES=20D8   SS=20D8   CS=20D8   IP=0110     NV UP EI PL NZ NA PO NC
20D8:0110 EBF4            JMP     0106
-t

AX=00F1   BX=0058   CX=0000   DX=0000   SP=FFEE   BP=0000   SI=0000   DI=0000
DS=20D8   ES=20D8   SS=20D8   CS=20D8   IP=0106     NV UP EI PL NZ NA PO NC
20D8:0106 833F00          CMP     WORD PTR [BX],+00                   DS:0058=0000
-t

AX=00F1   BX=0058   CX=0000   DX=0000   SP=FFEE   BP=0000   SI=0000   DI=0000
DS=20D8   ES=20D8   SS=20D8   CS=20D8   IP=0109     NV UP EI PL ZR NA PE NC
20D8:0109 7407            JZ      0112
-t

AX=00F1   BX=0058   CX=0000   DX=0000   SP=FFEE   BP=0000   SI=0000   DI=0000
DS=20D8   ES=20D8   SS=20D8   CS=20D8   IP=0112     NV UP EI PL ZR NA PE NC
20D8:0112 8907            MOV     [BX],AX                             DS:0058=0000
-t

AX=00F1   BX=0058   CX=0000   DX=0000   SP=FFEE   BP=0000   SI=0000   DI=0000
DS=20D8   ES=20D8   SS=20D8   CS=20D8   IP=0114     NV UP EI PL ZR NA PE NC
20D8:0114 90              NOP
-d 0050 005f
20D8:0050   30 00 15 00 A0 00 0C 00-F1 00 00 00 00 20 20 20   0............
-
            Datos originarios       Suma
```

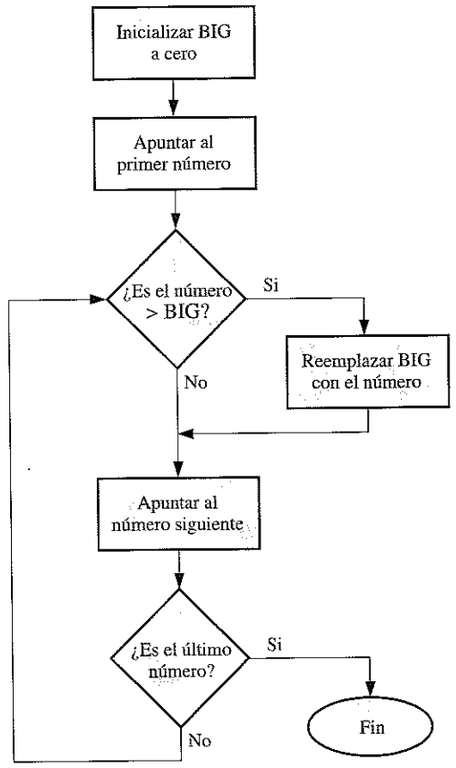
FIGURA 14.17
Última parte del trazado del programa de suma. El resultado de la suma, 00F1, se muestra en la última línea, con la parte menos significativa (F1) en primer lugar.

Continuando este proceso, puede ejecutarse el código completo y observar los cambios en los registros a medida que el microprocesador ejecuta las instrucciones, como se muestra en la Figura 14.17. Este programa tiene una estructura de programación común, denominada *bucle*. Un bucle es un grupo de instrucciones que se ejecuta repetidas veces hasta que se cumple una determinada condición; en este caso, la condición es encontrar un dato que sea igual a cero. Una vez encontrado el valor cero, se ejecutará la última instrucción y se almacenará la suma (00F1 es la suma en hexadecimal) en lugar del cero que indicaba el último dato. Puede verse esto escribiendo **d 0050 005F** (mostrar direcciones entre 0050 y 005F) en el indicativo de Debug cuando llegue a la última instrucción (NOP), como muestra la Figura 14.17. El resultado aparece en los bytes de las posiciones novena y décima (la quinta palabra), en la línea que sigue a la instrucción presentada en pantalla. Obsérvese que se muestra en primer lugar la parte menos significativa del resultado. Cuando el microprocesador realiza la ejecución "en tiempo real", realmente este programa no necesita más de 1 µs para completar el proceso. Si decide repetir el proceso, tendrá que volver a cargar el valor cero en la posición 0058, dado que ha sido reemplazado por la suma. Recuérdese que el programa utiliza el cero como un detector de "fin de datos".

EJEMPLO 14.2 Escribir un programa en lenguaje ensamblador que busque el mayor número sin signo entre los datos y lo coloque en la última posición. Suponer que el último dato se indica mediante un cero.

Solución. El organigrama se muestra en la Figura 14.18.

FIGURA 14.18
Organigrama. La variable BIG representa el valor mayor.



Se supone que los datos son los mismos que antes. El listado del programa, incluyendo los comentarios, sería el siguiente:

```
        mov ax, 0000          ; el valor inicial de BIG se encuentra
                              ; en el registro ax
        mov bx, 0050          ; apuntar a una posición de memoria (50H)
                              ; donde comiencen los datos
repeat: cmp [bx], ax          ; ¿es el dato mayor que BIG?
        jbe check             ; si el dato es menor, ir a
                              ; "check",
        mov ax, [bx]          ; en caso contrario, almacenar en ax el
                              ; dato mayor
check: add bx,02              ; apuntar al número siguiente en memoria
                              ; (dos bytes por palabra)
       cmp word ptr [bx],0    ; comprobar si es el último dato
       jnz repeat             ; continuar si el dato no es cero
       mov [bx], ax           ; guardar BIG en memoria
       nop                    ; no realiza ninguna operación.
```


En la Figura 14.19 se muestra el listado en Debug de este programa. Los datos se introducen de la misma forma que antes, comenzando en la posición 0050. En este caso, se usan los mismos datos, pero, si se prefiere, se puede elegir nuevos datos. Es importante que se introduzca un cero en la última posición de datos, ya que el programa continúa hasta que encuentra dicha posición. El cero se reemplaza, cada vez que el programa se ejecuta, con el dato de mayor valor. El programa se introduce iniciando el ensamblador en la posición 100, introduciendo el programa después de ejecutar el comando **a 100**.

FIGURA 14.19
Listado en Debug de una parte del programa.

```
-a 100
20D8:0100 mov ax,0
20D8:0103 mov bx,50
20D8:0106 cmp [bx],ax
20D8:0108 jbe 10c
20D8:010A mov ax, [bx]
20D8:010C add bx,2
20D8:010F cmp word ptr [bx],0
20D8:0112 jnz 106
20D8:0114 mov [bx],ax
20D8:0116 nop
20D8:0117
```

Se puede hacer el trazado del programa para ver su ejecución paso a paso. Alternativamente, puede introducirse el comando **g = 100 116** para ejecutar todas las instrucciones comprendidas entre las direcciones 100 y 116. La Figura 14.20 muestra los datos antes y después de la ejecución. Obsérvese que cada dato se almacena en dos bytes (aunque los datos tienen una longitud de un byte), ya que se han definido como palabras. Obsérvese también que el byte de menor peso precede al de mayor peso en memoria. Después de ejecutar el programa, la última posición de datos (que antes era cero) debe contener el valor mayor (A0 en este ejemplo). Puede comprobarse que este valor se encuentra tanto en AX como en memoria.

```
-a 100
20D8:0100 mov ax,0
20D8:0103 mov bx,50
20D8:0106 cmp [bx],ax
20D8:0108 jbe 10c
20D8:010A mov ax, [bx]
20D8:010C add bx,2
20D8:010F cmp word ptr [bx],0
20D8:0112 jnz 106
20D8:0114 mov [bx],ax
20D8:0116 nop
20D8:0117                                  Fin de señal de datos
-d 50 5f
20D8:0050   30 00 15 00 A0 00 0C 00-00 00 00 00 00 20 20 20    0............
-g= 100 116
            Datos antes de ejecutar el programa
AX=00A0  BX=0058  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=20D8  ES=20D8  SS=20D8  CS=20D8  IP=0116    NV UP EI PL ZR NA PE NC
20D8:0116 90              NOP
-d 50 5f
20D8:0050   30 00 15 00 A0 00 0C 00-A0 00 00 00 00 20 20 20    0............
-
-
-
-
            Los datos no cambian
                                    Ahora el fin de datos
                                    contiene el dato mayor
```

FIGURA 14.20
Datos antes y después de la ejecución.

Problema relacionado. Explicar cómo se puede modificar el organigrama para encontrar el número más pequeño de la lista, en lugar del más grande.

Tipos de instrucciones

Los programas de esta sección sólo han mostrado unas pocas de los centenares de variaciones de instrucciones disponibles para los programadores. Para simplificar el aprendizaje del juego de instrucciones de Intel, las instrucciones se dividen en siete categorías, que se describen a continuación:

Transferencia de datos. La instrucción de transferencia de datos más sencilla es MOV y se ha visto en los programas de ejemplo. Por ejemplo, la instrucción MOV se puede utilizar de varias formas para copiar un byte, una palabra (16 bits) o una doble palabra (32 bits) entre varios orígenes y destinos, como son registros, posiciones de memoria y puertos de E/S. Es posible que un mejor mnemónico para MOV hubiera sido "COPY", ya que es esto, copiar, lo que la instrucción realmente hace. Otras instrucciones de transferencia de datos son: IN (extrae datos de un puerto), OUT (envía datos a un puerto), PUSH (copia datos en la pila, que es un área separada de la memoria), POP (copia datos desde la pila) y XCHG (exchange, intercambio).

Aritméticas. Existe una serie de instrucciones y variantes para la suma, sustracción, multiplicación y división. La instrucción ADD se ha utilizado en ambos programas de ejemplo. Otras instrucciones de este tipo son: INC (incremento), DEC (decremento), CMP (comparación), SUB (sustracción), MUL (multiplicación) y DIV (división). Las variantes de estas instrucciones permiten realizar operaciones de acarreo y operaciones aritméticas con signo y sin signo. Estas instrucciones permiten la especificación de operandos localizados en memoria, en los registros y en los puertos de E/S.

Manipulación de bits. Este grupo de instrucciones incluye aquellas que se emplean para tres tipos de operaciones: lógicas (booleanas), de desplazamiento y de rotación. Las instrucciones lógicas son NOT, AND, OR, XOR y TEST. Un ejemplo de instrucción de desplazamiento es SAR (desplazamiento aritmético a la derecha). Un ejemplo de instrucción de rotación es ROL (rotación a la izquierda). Cuando los bits de un operando se desplazan en un registro y salen de él se pierden, pero cuando rotan se realimentan por el otro extremo. Estas instrucciones lógicas, de desplazamiento y de rotación pueden trabajar con bytes o con palabras de los registros o de la memoria.

Bucles y saltos. Estas instrucciones se utilizan para alterar la secuencia normal (una detrás de otra) de las instrucciones. La mayor parte de este tipo de instrucciones comprueba los indicadores del procesador para determinar qué instrucción debería procesarse a continuación. En el Ejemplo 14.2, las instrucciones JBE y JNZ se emplearon para alterar la secuencia de procesamiento. Otras instrucciones de este grupo son JMP (salto incondicional), JA (salto cuando se excede un valor), JO (salto de desbordamiento), LOOP (decrementa el registro CX y repite si es distinto de cero) y muchas otras.

Cadenas. Una **cadena** es una secuencia **contigua** de bytes o palabras (un elemento detrás de otro). Las cadenas son muy habituales en los programas de computadora.

Un ejemplo simple sería el de una frase que el programador quisiera mostrar en la pantalla. Hay disponibles cinco instrucciones básicas para cadenas, que sirven para copiar, cargar, almacenar, comparar o explorar una cadena, bien sea un byte cada vez o una palabra cada vez. Ejemplos de instrucciones de cadenas son MOVSB (copia una cadena, un byte cada vez) y MOVSW (copia una cadena, una palabra cada vez).

Subrutinas e interrupciones. Una **subrutina** es un miniprograma que se puede utilizar de forma repetida, pero que sólo se necesita escribir una vez. Por ejemplo, si un programador necesita convertir números ASCII procedentes de las pulsaciones en un teclado a formato BCD, una forma simple de estructurar el programa consiste en convertir las instrucciones necesarias en un proceso separado y "llamar" al proceso cuando sea necesario. Las instrucciones de este grupo incluyen el comando CALL (inicio de la subrutina) y RET (retorno al programa principal).

Control del procesador. Es un pequeño grupo de instrucciones que permite el control directo de los indicadores del procesador y otras tareas diversas. Un ejemplo es la instrucción STC (set carry flag, activar indicador de acarreo).

REVISIÓN DE LA SECCIÓN 14.4

1. Definir el término *programa.*
2. ¿Qué es un juego de instrucciones complejo?
3. ¿Qué es un código de operación?
4. ¿Qué es un operando?
5. ¿Qué es una cadena?

14.5 ■ LA UNIDAD CENTRAL DE PROCESO (CPU)

Una unidad central de proceso (CPU, Central Proccesing Unit) típica está formada por un microprocesador y varios circuitos integrados asociados, denominados circuitos de soporte. En primer lugar, vamos a ver el funcionamiento general de una CPU, y luego examinaremos una implementación específica utilizando el microprocesador 8088 como modelo básico, mencionando algunas de sus diferencias con los procesadores más recientes.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar el propósito de la CPU en una computadora. ■ Conocer los buses del sistema. ■ Describir una CPU básica que utilice un microprocesador 8088. ■ Explicar el propósito de un controlador de bus. ■ Describir los mecanismos de implementación de la interfaz con los buses.

Como hemos visto en la última sección, las instrucciones de la familia 80X86 tienen que ser compatibles en sentido ascendente, para evitar el problema de que los programas antiguos pasen a ser obsoletos de forma inmediata cuando surge un nuevo procesador. Motorola ha hecho lo mismo con su familia 68000. Las instrucciones del 8086/8088 pueden ejecutarse en los nuevos procesadores (pero no a la inversa). Aunque el procesador original 8086/8088 hace tiempo que es obsoleto para los PC,

todavía hay muchas tarjetas controladoras que utilizan dicho procesador y los circuitos integrados microcontroladores asociados. Además, muchas de las mejoras significativas que se han producido han tenido repercusión sobre la velocidad del procesador (como por ejemplo la frecuencia de la señal de reloj y las técnicas de pipeline), pero estas mejoras han tenido un efecto relativamente pequeño sobre la estructura básica de programación; una excepción a esto es la adición del modo protegido que resulta bastante complejo. Por estas razones, un estudio del procesador 8086/8088 y de los circuitos integrados de soporte será un buen punto de partida para entender la arquitectura básica de las computadoras. Comenzaremos este estudio presentando los procesadores 8086/8088 y destacaremos algunas mejoras clave que se han incorporado con los nuevos procesadores, para proporcionar una idea de los cambios tecnológicos más importantes.

CPU basada en el 8088

Una CPU (o MPU) típica se implementa mediante un microprocesador y algunos circuitos de soporte adicionales. La Figura 14.21 muestra una CPU basada en un 8088.

El microprocesador 8086/8088. Como ya sabemos, el 8088 es básicamente igual al microprocesador 8086, excepto en que el 8088 tiene un bus externo de datos de 8 bits, en lugar de uno de 16 bits. El 8088, con su bus de datos de 8 bits, es compatible con muchos dispositivos periféricos de 8 bits. Sin embargo, el 8088 puede manejar internamente los datos en palabras de 16 bits o en bytes, dependiendo de cómo se haya programado. También dispone, al igual que el 8086, de un bus de direcciones de 20 bits, que le permite utilizar hasta 1 Mbyte de memoria.

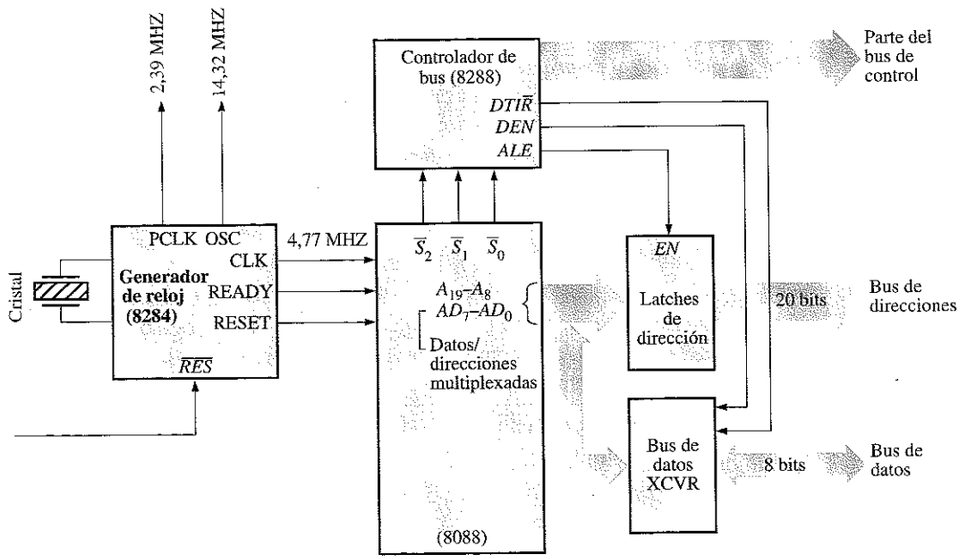


FIGURA 14.21
CPU simplificada basada en un 8088.

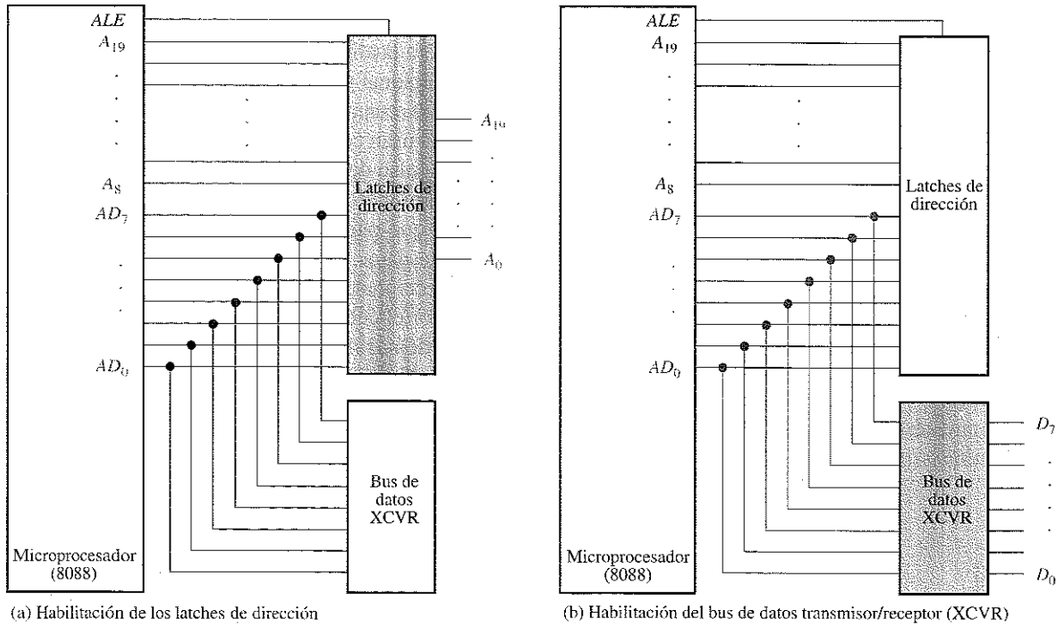
En los procesadores 8086/8088, el bus de datos está combinado con las líneas de dirección más bajas. Existe un total de 20 líneas que se emplean para el direccionamiento; en el 8088, estas líneas son AD_0–AD_7 y A_8–A_{19}. Las ocho líneas designadas mediante AD_0–AD_7 sirven como líneas de datos y de direcciones, utilizando un modo de operación multiplexado. Las líneas designadas mediante A_8–A_{19} se usan estrictamente para direccionamiento. En el 8086, las 16 líneas más bajas (AD_0–AD_{15}) se multiplexan; A_{16}–A_{19} se usan estrictamente para direccionamiento. Los procesadores 80186/80188 también utilizan líneas de datos y de dirección, aunque éstos son los únicos procesadores de la familia 80X86 que multiplexan las líneas de datos y de dirección. La principal ventaja de este método era la eliminación de pines, pero, a cambio, estos procesadores requieren circuitería adicional para demultiplexar dichas líneas.

Como se ilustra en la Figura 14.22(a), cuando el microprocesador 8088 envía una dirección de 20 bits, las veinte líneas se usan como bus de direcciones. Cuando se envía la dirección, también se envía un impulso denominado *ALE* (address latch enable, habilitación del latch de dirección) para indicar a la circuitería externa que hay una dirección estable en las líneas *AD* y que puede ser introducida en los latches. Cuando el microprocesador envía o recibe datos, se usan las ocho líneas de menor peso, AD_0–AD_7 , como bus de datos bidireccional, como muestra la Figura 14.22(b). La multiplexación del bus está regulada por el controlador de bus. El controlador de bus es uno de los circuitos de soporte. Las demás líneas de entrada y salida del microprocesador se describirán junto con los circuitos de soporte correspondientes.

El generador de reloj. Un circuito generador de reloj, tal como el 8284, proporciona las señales de temporización básicas al 8088 y a otras partes del sistema, incluyendo los periféricos. También proporciona la señal de RESET al 8088 para inicializar la circuitería interna, y la señal de READY para sincronizar el microprocesador con el resto del sistema. Se utiliza un cristal resonante externo para establecer la frecuencia de oscilación (en este caso, 14,31818 MHz), como muestra la Figura 14.23. Esta frecuencia está disponible en la salida OSC. Un contador divisor por tres interno del generador de reloj produce una señal de reloj a 4,772727 MHz con un ciclo de trabajo del 33% para excitar al microprocesador. Esta señal está disponible en la salida CLK. También se obtiene una señal de 2,386363 MHz a partir de la frecuencia CLK, la cual está disponible en la salida PCLK del generador de reloj. EL 8086/8088 ha sido el único procesador que utilizó el generador de reloj 8284; sin embargo, el 80286 utilizaba un circuito integrado similar, el 82284. Los procesadores más recientes incorporan internamente muchas de las funciones del generador de reloj con el fin de reducir el número de componentes requeridos por el sistema. El Pentium requiere un oscilador externo que funcione a la misma frecuencia (actualmente, varios cientos de MHz) que el sistema para controlar la velocidad básica del sistema.

El controlador de bus. El controlador de bus 8288 se usa para relevar al microprocesador 8088 de la mayor parte de las funciones de control del bus, de modo que se maximice la eficiencia del procesamiento. Una de las funciones del controlador de bus en la CPU es proporcionar las señales para multiplexar las líneas de bus AD_0–AD_7 procedentes del microprocesador. El controlador de bus genera estas señales de control basándose en los códigos de las líneas de estado $\overline{S}_0$, $\overline{S}_1$ y $\overline{S}_2$ del microprocesador. Cuando el microprocesador necesita comunicarse con la memoria o la E/S, envía un código por las líneas de estado al controlador de bus. Entonces, el controlador de bus genera una señal *ALE*, que permite que la dirección que se encuentra en *las líneas*

AD_0–AD_7 y A_8–A_{19} del microprocesador se almacene en los latches de dirección. Existen 20 latches (A_0–A_{19}), cuyas salidas de tres estados forman el bus de direcciones del sistema. La Figura 14.24 ilustra los pasos de esta operación. Durante el ciclo de lectura/escritura de la memoria, hay disponible una dirección válida en el bus de direcciones y, por tanto, los ocho bits de menor peso (AD_0–AD_7) del bus combinado del microprocesador quedan libres para enviar o recibir datos.



(a) Habilitación de los latches de dirección

(b) Habilitación del bus de datos transmisor/receptor (XCVR)

FIGURA 14.22
Funcionamiento del bus multiplexado del 8088.

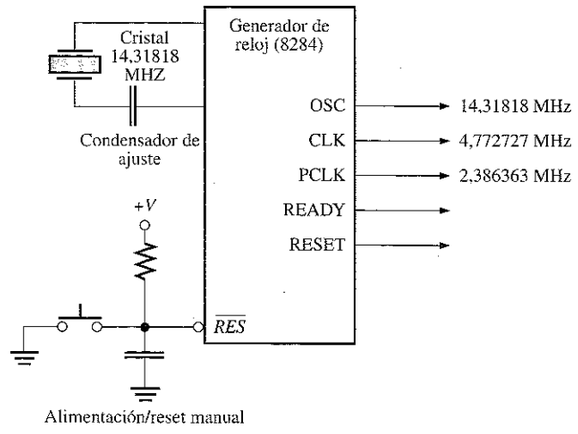


FIGURA 14.23
El generador de reloj 8284.

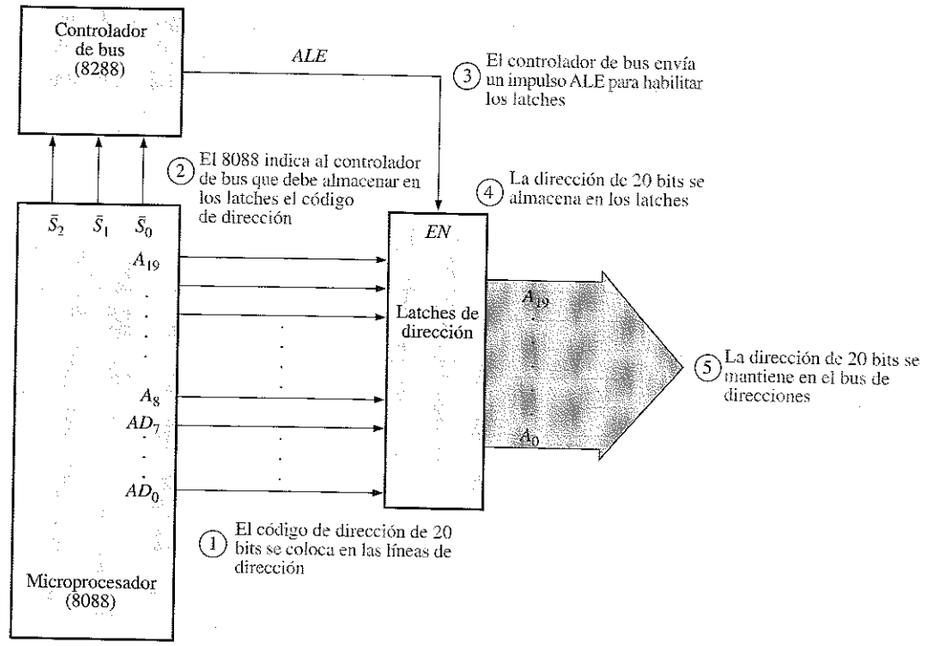


FIGURA 14.24
Pasos para colocar una dirección válida en el bus.

Después de haber almacenado la dirección en los latches, el microprocesador comunica al controlador de bus, a través de las líneas de estado ($\bar{S}_0$, $\bar{S}_1$ y $\bar{S}_2$), que quiere leer o escribir datos en la posición de memoria o de E/S especificada por la dirección que se encuentra en el bus de direcciones. Los códigos de las líneas de estado para las cuatro posibles operaciones de lectura/escritura se facilitan en la Tabla 14.1.

TABLA 14.1
Códigos de estado de lectura/escritura del 8088.

Códigos de estado			*Operación*	*Comando de salida del controlador de bus activo*
$\bar{S}_2$	$\bar{S}_1$	$\bar{S}_0$		
0	0	1	Lectura del puerto de E/S	$\overline{IORC}$
0	1	0	Escritura del puerto de E/S	$\overline{IOWC}$
1	0	1	Lectura de memoria	$\overline{MRDC}$
1	1	0	Escritura en memoria	$\overline{MWTC}$

Cuando el controlador de bus recibe uno de los códigos de estado, envía un comando de lectura ($\overline{IORC}$ o $\overline{MRDC}$) o un comando de escritura ($\overline{IOWC}$ o $\overline{MWTC}$) al dispositivo apropiado, y también envía dos señales al transmisor/receptor de tres estados del bus de datos (*XCVR*). Estas dos señales son *DT/*$\overline{R}$ (Data transmit/$\overline{\text{receive}}$,

transmisión/recepción de datos) y *DEN* (data enable, habilitación de datos). La señal *DEN* habilita el transmisor/receptor del bus de datos, y la señal *DT/$\overline{R}$* selecciona la dirección de los datos en el bus de datos (D_0 a D_7). La Figura 14.25 ilustra una operación de lectura de memoria en el 8088 y la Figura 14.26 una operación de escritura en memoria en dicho microprocesador. Los cronogramas de los ciclos de lectura y escritura se facilitan en la Sección 14.6.

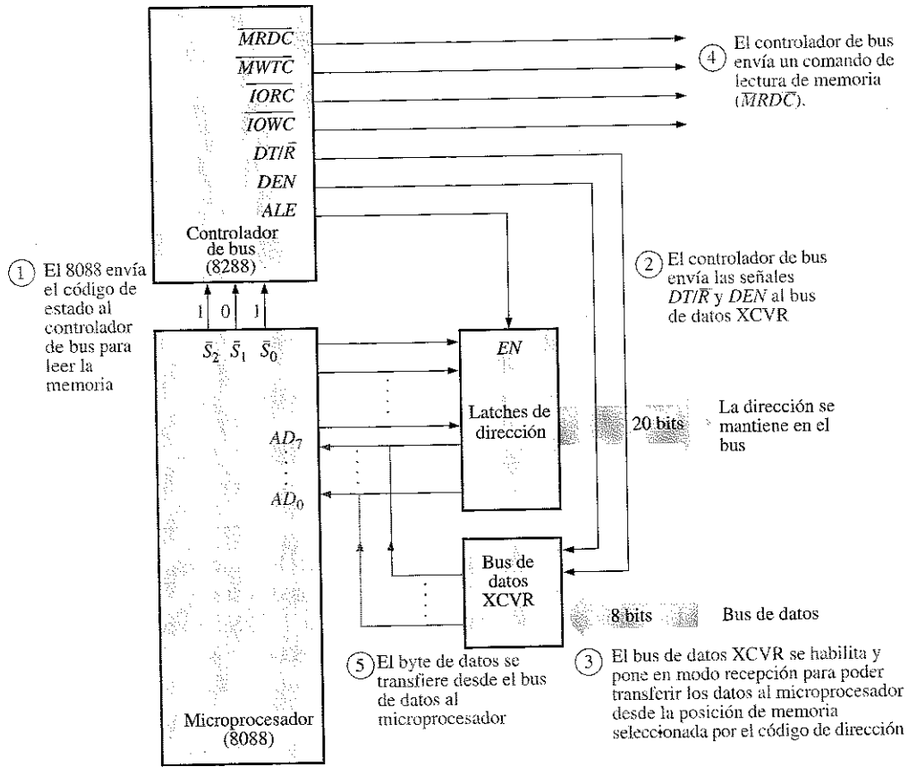


FIGURA 14.25
Ejemplo de una operación de lectura de memoria en un 8088.

Aunque se han añadido muchas señales a los nuevos microprocesadores para realizar tareas diversas (como, por ejemplo, el control de caché), otras señales de interfaz no han cambiado. El tipo de ciclo de bus y las señales de interrupción (con una adición) son prácticamente las mismas que en el 8086/8088.

La serie de procesadores que se inició con el 80286 no multiplexa la dirección ni los datos y, por tanto, no requiere circuitos latch de dirección, aunque el número de pines se aumentó y fue necesario un nuevo encapsulado. Originalmente, para el 8086/8088 se utilizó un encapsulado de 40 pines, en dos filas. Los nuevos procesadores pueden tener varios cientos de pines; el Pentium Pro, por ejemplo, tiene 387. La

tecnología de encapsulado ha mejorado enormemente en la última década, incluyendo un incremento en el número de pines, un mejor rendimiento térmico y menores tamaños, permitiendo una densidad de componentes significativamente mayor.

Mejoras en los procesadores

Un cambio importante en los procesadores desde el 8086/8088 es el uso de la arquitectura pipeline. Las pipelines aceleran el procesamiento de las instrucciones, extrayendo y ejecutando instrucciones en varias etapas, todas las cuales trabajan en paralelo. Algunas instrucciones pueden ejecutarse en un único ciclo de reloj en el Pentium, mientras que en el 8086/8088 tardaban 16 ciclos de reloj o más.

Una segunda mejora de importancia es la adición y expansión de la memoria caché. Esta memoria puede ser interna o externa al procesador. Una memoria caché interna está diseñada para ser muy rápida. Puede almacenar instrucciones o datos y aumentar la velocidad del procesador, eliminando los retardos requeridos para acceder a la memoria externa. El Pentium original fue el primer procesador que incorporó una caché de datos y una caché de instrucciones separadas.

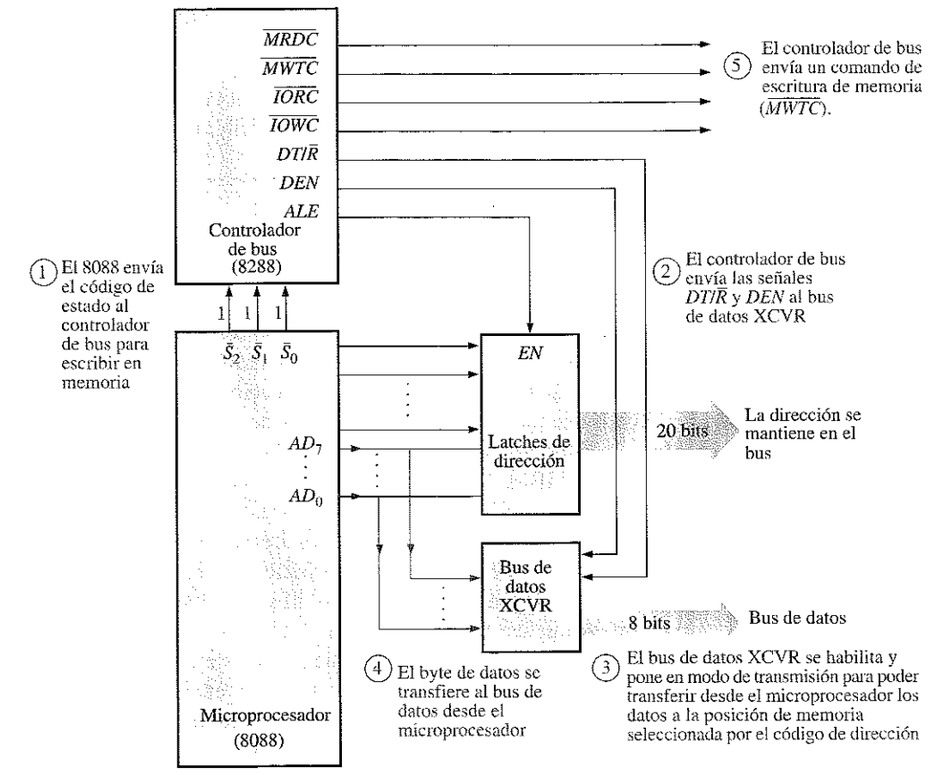


FIGURA 14.26
Ejemplo de operación de escritura en la memoria de un 8088.

La tercera mejora es la inclusión del coprocesador matemático interno. Este dispositivo ejecuta las funciones matemáticas mucho más rápidamente y de forma más eficiente que los coprocesadores del 80486 y posteriores. Además, se han añadido al Pentium un doble procesador interno para operaciones con enteros y una lógica de predicción de saltos. Éstas y otras varias mejoras han cambiado los requisitos para la lógica de control de bus, que ahora se implementa con dispositivos PLD.

REVISIÓN DE LA SECCIÓN 14.5

1. ¿Cuál es el propósito del generador de reloj en la CPU?
2. Enumerar dos funciones del controlador de bus.
3. Explicar por qué es necesaria la multiplexación de bus en una CPU basada en un 8088.
4. ¿Qué tres mejoras del Pentium no estaban disponibles en el 8086/8088?

14.6 ■ LA MEMORIA

Esta sección cubre la parte de la memoria de una computadora típica y la forma en que la utiliza la CPU.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar la asignación de direcciones en las memorias de las computadoras. ■ Describir las operaciones de lectura y escritura. ■ Describir un ciclo de lectura de memoria. ■ Describir un ciclo de escritura en memoria.

Asignación de direcciones

Como ya hemos visto, la CPU 8086/8088 puede direccionar hasta 1.048.576 posiciones con un bus de direcciones de 20 bits (A_0–A_{19}). En una computadora típica, una parte del espacio total de memoria se asigna a la RAM y la ROM, y otra parte a la E/S. Los puertos de entrada y salida tienen sus propias direcciones exclusivas (dedicadas) en los procesadores Intel, aunque utilizan posiciones de memoria normales en otros procesadores. Esto se estudiará con más detalle en la Sección 14.7.

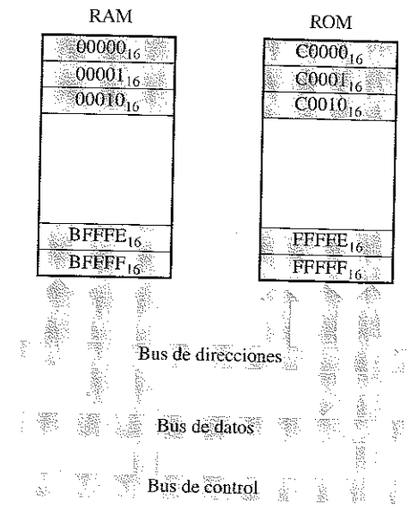
Los diseñadores del IBM PC original dividieron la cantidad total de 1 Mbyte de memoria RAM y memoria ROM en tres áreas generales: un área de usuario, un área de vídeo y un área de sistema. Dentro de estas tres áreas, se definieron ciertas posiciones reservadas para propósitos específicos. La parte inferior de la RAM (denominada memoria convencional) era un bloque de 640 kbytes que comenzaba en la posición 0 de la memoria y que estaba disponible principalmente para los usuarios (una pequeña parte estaba reservada para que la empleara el sistema). En esa época, esto era una gran cantidad de memoria de usuario (puesto que era diez veces más que la memoria de la que disponían los procesadores de 8 bits), aunque para los estándares actuales sea muy pequeña. Recuérdese que el Pentium puede direccionar 4 Gbytes.

Inmediatamente encima del bloque de 640 kbytes de RAM, se reservó un bloque de 128 kbytes para la RAM de vídeo, entre las direcciones A0000 y BFFFF. Este área fue reservada para la información que se muestra en la pantalla.

En la parte superior de la memoria de 1 Mbyte disponible, se asignó un bloque de 256 kbytes (que comienza en la dirección C0000 y termina en FFFFF), para los programas del sistema y se implementó con una ROM. La última parte de esta ROM incluía el BIOS (Basic Input Output System). Cuando se reiniciaba, el sistema accedía a una serie de posiciones de memoria del BIOS para "arrancar" el sistema. El sistema BIOS también reservaba una pequeña sección de la RAM (entre las direcciones 400 y 4FF) como área de trabajo, y el sistema utilizaba otras partes de la RAM próximas a la parte inferior de la memoria para almacenar determinada información necesaria. La mayor parte de esta estructura se ha mantenido en todos los microprocesadores Intel.

La Figura 14.27 muestra un mapa básico de memoria de una computadora 8086/8088 típica. La RAM tiene asignadas a las direcciones comprendidas entre 00000_{16} y $BFFFF_{16}$ (un total de 786.431 posiciones), que incluyen las áreas de vídeo y RAM que hemos mencionado. La ROM tiene asignadas las direcciones $C0000_{16}$ a $FFFFF_{16}$ (un total de 262.143 posiciones).

FIGURA 14.27
Ejemplo de asignación típica de direcciones de memoria en una computadora 8086/8088.



Cuando el número de líneas de dirección aumentó en los subsiguientes procesadores, se conservó gran parte de la organización anterior de memoria; sin embargo, se introdujeron algunos cambios significativos. Con el 80386, se añadió un nuevo mecanismo de *paginación* al procesador para el acceso a memoria. Esta unidad formaba parte de una unidad de gestión de memoria (MMU) que tenía sus raíces en el microprocesador 80286. La paginación permitía ubicar a los programas en cualquier lugar en que hubiera espacio disponible y que éstos continuaran funcionando adecuadamente mediante la traducción de las direcciones. Esto significa que un programa que necesitaba acceder a una determinada ubicación podía funcionar incluso aunque dicha ubicación estuviera ocupada por otro programa. Para mantener la compatibilidad con los procesadores anteriores, la memoria situada por encima de la dirección FFFFF se denomina memoria extendida. La memoria que se encuentra entre las direcciones

00000 y FFFFF se conoce como área de memoria real. Los programas que funcionan en modo real emplean este área.

Operación CPU-memoria

La CPU realiza dos tipos de transferencia de memoria, *lectura* y *escritura*. Básicamente, durante la operación de lectura, la CPU carga una instrucción de programa, otra dirección (que se presenta como un dato) o un dato. Durante la operación de escritura, la CPU envía los datos resultantes de un cálculo o de otra operación a la RAM. En la Figura 14.28 se presenta un diagrama de la CPU 8088 y de la memoria de una computadora.

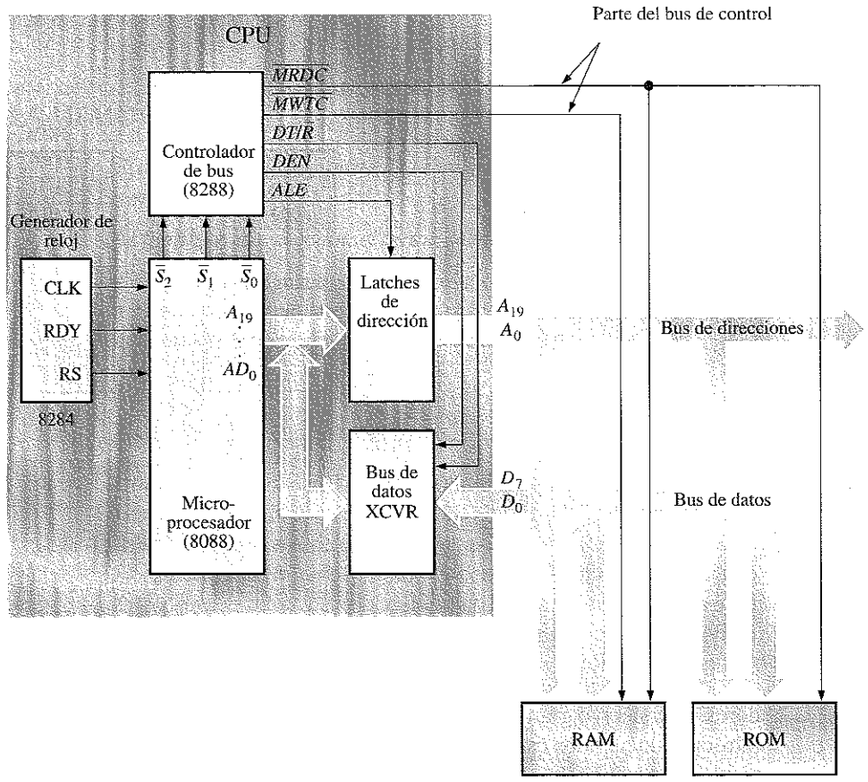


FIGURA 14.28
Organización de la memoria en una CPU 8088.

El ciclo de lectura de memoria. El microprocesador 8086/8088 inicia un ciclo de lectura de memoria enviando un código de estado, $\overline{S}_2\,\overline{S}_1\,\overline{S}_0 = 101$, al controlador del bus y colocando una dirección binaria en las líneas del bus AD_0–AD_7 y A_8–A_{19}. El controlador de bus genera entonces una señal *ALE,* que almacena en el latch la dirección

que se encuentra en el bus de direcciones (A_0–A_{19}), dejando libres las líneas AD_0–AD_7 para transferir los datos. El controlador de bus también genera una señal $DT/\overline{R}$ a nivel bajo para que el transmisor/receptor del bus se configure para recibir los datos de la memoria y pasarlos al microprocesador.

A continuación, el controlador de bus genera un comando de lectura de memoria ($\overline{MRDC}$) que envía a la memoria. Éste hace que la memoria coloque el contenido de la dirección seleccionada en el bus de datos. La señal *DEN* procedente del controlador del bus habilita al transmisor/receptor de bus, y un byte de datos fluye hacia el microprocesador. La Figura 14.29 muestra el diagrama de tiempos básico de un ciclo de lectura de memoria del 8088.

Ciclo de escritura en memoria. El microprocesador inicia un ciclo de escritura en memoria enviando el código de estado $\overline{S}_2\,\overline{S}_1\,\overline{S}_0 = 110$ al controlador del bus y colocando una dirección binaria en las líneas del bus. Entonces, el controlador de bus emite una señal *ALE* para almacenar en el latch la dirección que está en el bus de direcciones (A_0–A_{19}) y una señal $DT/\overline{R}$ a nivel alto para configurar el transmisor/receptor para el envío de datos a la memoria. A continuación, el microprocesador coloca el byte de datos en las líneas AD_0–AD_7. La señal *DEN* procedente del controlador del bus habilita al transmisor/receptor del bus para colocar el byte de datos en el bus de datos (D_0–D_7).

Después, el controlador de bus envía a la memoria un comando de escritura ($\overline{MWTC}$). Este comando hace que la RAM almacene el byte de datos en la dirección seleccionada. La Figura 14.30 muestra un cronograma básico para un ciclo de escritura en memoria.

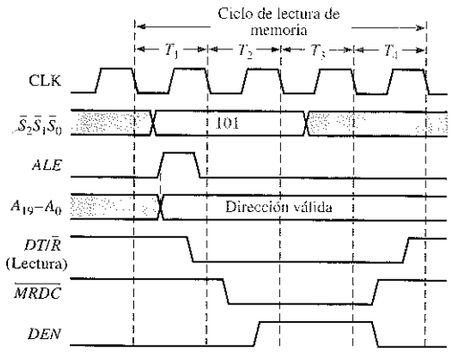


FIGURA 14.29
Cronograma básico de un ciclo de lectura de memoria basado en el 8088.

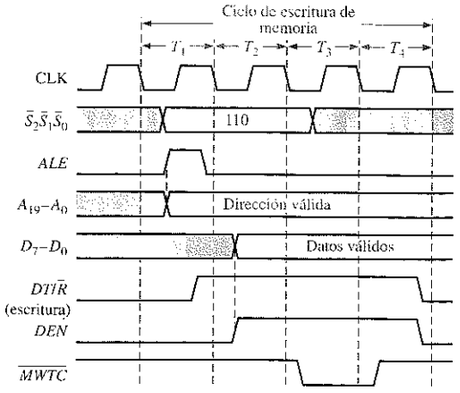


FIGURA 14.30
Cronograma básico de un ciclo de escritura de memoria basado en el 8088.

REVISIÓN DE LA SECCIÓN 14.6

1. Nombrar los dos ciclos que utiliza la CPU 8088 para transferir datos a y desde la memoria.
2. ¿Cuántos ciclos de reloj se utilizan para leer datos de la memoria, en un sistema basado en un 8088?
3. ¿Cuándo se presenta una dirección de memoria válida en el bus de direcciones (A_0–A_{19}) durante el ciclo de escritura?

14.7 ■ LOS PUERTOS DE ENTRADA/SALIDA (E/S)

En esta sección veremos de forma general cómo se comunica la CPU con los dispositivos de entrada y salida a través de los puertos de E/S. Un puerto de E/S puede imaginarse como una "ventana" entre la computadora y el mundo exterior, a través de la cual pasa la información.

Al finalizar esta sección, el lector deberá ser capaz de:

- Explicar qué es un puerto de E/S.
- Comentar los tipos básicos de puertos.
- Explicar la diferencia entre puertos de E/S dedicados y puertos asignados a memoria.
- Indicar el propósito de una interfaz periférica programable.

En la Figura 14.31 se presenta un diagrama de bloques de un sistema de computadora básico, que incluye los puertos de E/S y los periféricos típicos. Obsérvese que un puerto puede ser estrictamente de entrada o estrictamente de salida, o bidireccional, es decir, de entrada y salida. Aunque en la figura sólo se muestran cuatro puertos de E/S, la CPU es capaz de gestionar muchos más. Todos los procesadores Intel 80X86 y Pentium pueden direccionar 64.000 puertos.

Como se ha dicho, un **puerto de E/S** proporciona una interfaz entre la computadora y el "mundo exterior" de los equipos periféricos. Dependiendo del diseño del sistema, los puertos de E/S pueden ser *dedicados* o *asignados a memoria*. Estos términos describen la forma en que la CPU accede a los puertos.

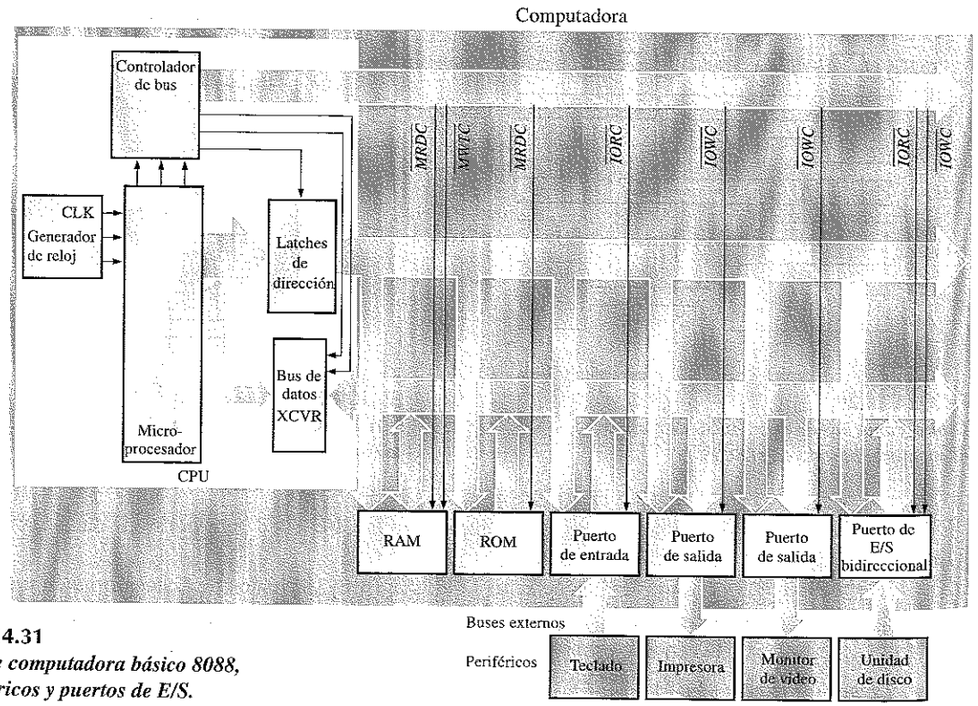


FIGURA 14.31
Sistema de computadora básico 8088, con periféricos y puertos de E/S.

Puertos de E/S dedicados

Un puerto de E/S dedicado tiene asignada una dirección exclusiva dentro del espacio de direcciones de E/S de la computadora, y para la comunicación se usan los comandos de E/S dedicados. Este espacio de direcciones está completamente separado de las direcciones de memoria. Los comandos de lectura y escritura de E/S, $\overline{IORC}$ e $\overline{IOWC}$ en un sistema basado en un 8086/8088, sirven para acceder a estos puertos.

Puerto de salida. El funcionamiento básico de un puerto de salida es el siguiente: el puerto ocupa una posición dentro del espacio de direcciones de E/S del sistema; es decir, tiene una dirección exclusiva, diferente de las direcciones de otros puertos. Los 16 bits de menor peso del bus de direcciones se usan para las direcciones de los puertos, ya que el espacio de E/S tiene un tamaño de 64k (2^{16} = 64k).

Durante un ciclo de escritura de la CPU, la dirección del puerto se coloca en el bus de direcciones, y la CPU envía una señal $\overline{IOWC}$ (I/O write command, comando de escritura de E/S) a nivel bajo para habilitar el decodificador de direcciones. El decodificador produce entonces una señal para almacenar el byte de datos en los latches correspondientes a las líneas de salida del puerto. La Figura 14.32 muestra cómo se puede implementar un puerto de salida dedicado.

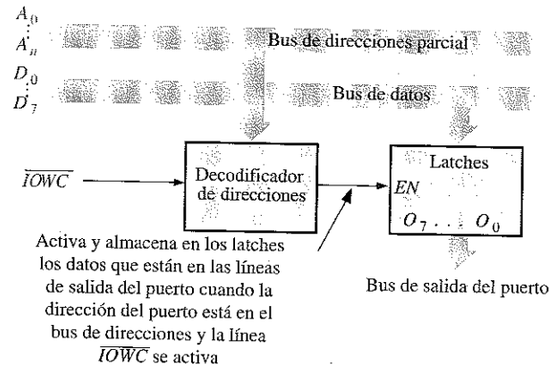


FIGURA 14.32
Puerto de salida dedicado básico.

Puerto de entrada. Un puerto de entrada dedicado no es tan sencillo como un puerto de salida, ya que las líneas de salida del puerto están conectadas al bus de datos del sistema. Esta disposición requiere que las líneas de salida del puerto se inhabiliten cuando el puerto no se utiliza, para evitar las interferencias con otras actividades en el bus de datos. La Figura 14.33 muestra un ejemplo de una implementación básica de un puerto de entrada.

Puerto bidireccional. El tercer tipo de puerto dedicado de E/S es el puerto bidireccional que, básicamente, es una combinación de los puertos de entrada y de salida, con algunos controles de activación adicionales. Un puerto bidireccional permite que los datos fluyan hacia o desde el dispositivo periférico, como en el caso de una unidad de disco.

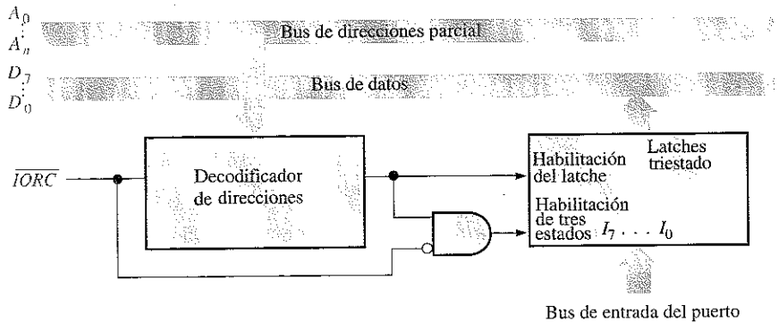


FIGURA 14.33
Puerto de entrada dedicado básico.

Puertos de E/S mapeados en memoria

Un segundo método para acceder a los puertos de E/S es el **puerto mapeado en memoria**. La implementación del puerto es esencialmente la misma que la de los puertos dedicados de las Figuras 14.32 y 14.33, excepto por dos cosas:

1. Los puertos asignados a memoria tienen asignadas direcciones dentro del espacio de direcciones de la memoria del ordenador y, realmente, la CPU los ve como posiciones de memoria. En el 8086/8088, para estos puertos se utiliza la totalidad de las veinte líneas de dirección.
2. La CPU accede a un puerto de E/S asignado a memoria mediante los comandos $\overline{MRDC}$ y $\overline{MWTC}$, en lugar de con $\overline{IORC}$ e $\overline{IOWC}$. En otras palabras, los puertos de E/S asignados a memoria se tratan exactamente igual que las posiciones de memoria. En lugar de utilizar las instrucciones IN y OUT en lenguaje ensamblador, para un puerto de E/S asignado a memoria se utiliza la instrucción MOV.

Aunque el diseño de la E/S asignado a memoria es muy simple, tiene la desventaja de utilizar espacio de memoria, con el riesgo potencial de conducir a una fragmentación de la memoria disponible. Además, la decodificación de direcciones requiere el bus de direcciones completo, en lugar de sólo una parte del mismo.

Interfaz periférica programable

En muchos sistemas se utilizan circuitos integrados especiales para implementar los puertos de E/S. Uno de estos dispositivos es la interfaz periférica programable (**PPI,** programmable peripheral interface) 8225A, que se muestra en la Figura 14.34. Esta interfaz PPI continúa siendo muy popular y se ha empleado en muchas aplicaciones de E/S y en tarjetas auxiliares que se conectan al bus de la computadora. El decodificador de direcciones es un dispositivo separado, que sirve para habilitar la PPI cuando se decodifica una dirección de un puerto. Esta PPI en particular dispone de tres puertos. Los puertos A y B tienen cada uno ocho líneas, y el puerto C se divide en dos grupos de 4 líneas. Los puertos se pueden programar para obtener diversas combinaciones de puertos de entrada, puertos de salida y puertos bidireccionales, así como

líneas de control y de establecimiento de comunicación, mediante la escritura de una palabra en un registro de control especial. Esto significa que el puerto se puede configurar dinámicamente, es decir, se puede modificar mediante software el tamaño y el tipo de puerto y se pueden añadir o eliminar secuencias de establecimiento de comunicación. La PPI se puede utilizar tanto para puertos dedicados como asignados a memoria.

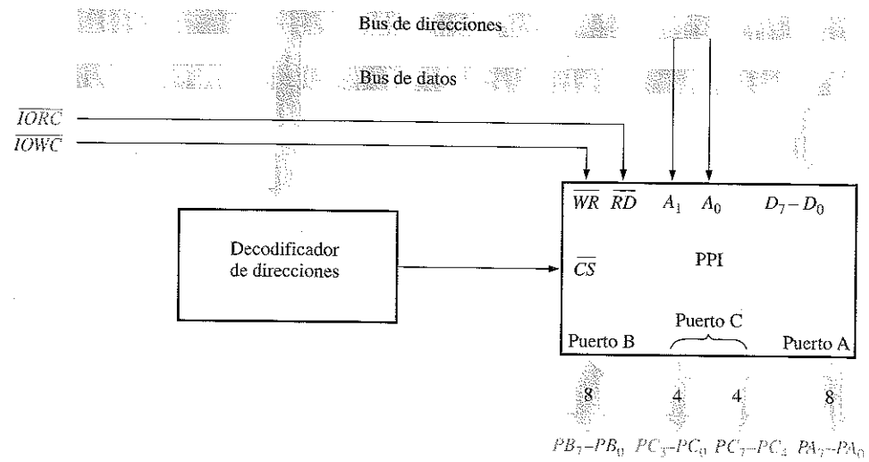


FIGURA 14.34
La interfaz periférica programable (PPI, programmable peripheral interface) 8255A.

REVISIÓN DE LA SECCIÓN 14.7

1. ¿Qué es un puerto de E/S?
2. Explicar la diferencia entre puerto de E/S dedicado y puerto de E/S asignado a memoria.
3. ¿Cuál es el propósito de la PPI?

14.8 ■ INTERRUPCIONES

El concepto de puerto de E/S se ha visto en la Sección 14.7, y se han presentado los dos tipos: puertos de E/S dedicados y asignados a memoria. En esta sección nos ocuparemos del establecimiento de la comunicación entre un periférico y la CPU. Se estudian tres métodos: la E/S por sondeo, la E/S por interrupción y las interrupciones software.

Al finalizar esta sección, el lector deberá ser capaz de:

- Explicar la necesidad de las interrupciones en un sistema de computadora.
- Describir el concepto básico de E/S por sondeo. ■ Describir el concepto básico de E/S por interrupción. ■ Explicar una interrupción software.

En un sistema basado en microprocesador, tal como una computadora personal, los dispositivos periféricos requieren que la CPU les dé servicio periódicamente. Generalmente, el término *servicio* significa el envío de datos o la recepción de datos desde el dispositivo, o la realización de algún proceso de actualización. Existen tres formas en que una rutina de servicio puede iniciarse: mediante la *E/S por sondeo*, la *E/S por interrupción* y las *interrupciones software*. Una **interrupción** es una señal o instrucción que hace que el proceso actual se detenga temporalmente mientras se ejecuta una rutina de servicio. En esta sección vamos a describir más en detalle los tres métodos mediante los que se puede prestar este servicio.

En general, los dispositivos periféricos son muy lentos comparados con la CPU. Por ejemplo, la CPU 8086/8088 puede realizar aproximadamente 1.000.000 de operaciones de lectura o escritura en un segundo (con una frecuencia de reloj de 4,77 MHz). Un impresora puede, por término medio, escribir unos pocos caracteres por segundo (un carácter se representa mediante ocho bits), dependiendo del tipo de información que esté imprimiendo y del tipo de impresora. Un teclado genera, aproximadamente, uno o dos caracteres por segundo, dependiendo de la velocidad del operador. Por tanto, entre los instantes en que la CPU debe atender a los periféricos, ésta puede llevar a cabo una gran cantidad de procesamiento, y en la mayoría de los sistemas este tiempo de proceso debe maximizarse utilizando un método eficiente para dar servicio a los periféricos.

E/S por sondeo

Uno de los métodos para dar servicio a los periféricos se denomina **sondeo** (polling). Con este método, la CPU debe comprobar secuencialmente cada dispositivo periférico a determinados intervalos, para ver si necesitan o están preparados para el servicio. La Figura 14.35 ilustra el método básico de la E/S por sondeo.

La CPU selecciona secuencialmente cada dispositivo periférico por medio del multiplexor, para ver si necesita servicio, comprobando el estado de una línea que indica si el dispositivo está preparado. Determinados periféricos pueden necesitar servicio a intervalos irregulares e impredecibles, es decir, más frecuentemente en unas ocasiones que en otras. Sin embargo, la CPU debe sondear al dispositivo a la velocidad más alta. Por ejemplo, supongamos que un determinado periférico necesita ocasionalmente servicio cada 1000 μs, pero la mayor parte de las veces requiere servicio sólo cada 100 ms. Como puede verse, se desperdicia un valioso tiempo de proceso si la CPU sondea al dispositivo, como debe hacerlo, a su velocidad máxima (cada 1000 μs), puesto que casi nunca el dispositivo va a necesitar servicio cuando se le sondea.

Cada vez que la CPU sondea a un dispositivo, debe detener el programa que está procesando, ir a la secuencia de sondeo, proporcionar el servicio en caso necesario y volver al punto en el que había dejado el programa en curso.

Otro problema del método de la E/S por sondeo secuencial es que, si dos o más dispositivos necesitan servicio a un mismo tiempo, el primero al que se sondee será el primero en recibir el servicio, y los demás deberán esperar, aunque precisen el servicio mucho más urgentemente que el primero seleccionado. Como puede verse, éste método sólo es deseable para dispositivos que precisen un servicio a intervalos regulares y predecibles, y únicamente en las situaciones en que no haya que tener en cuenta condiciones de prioridad.

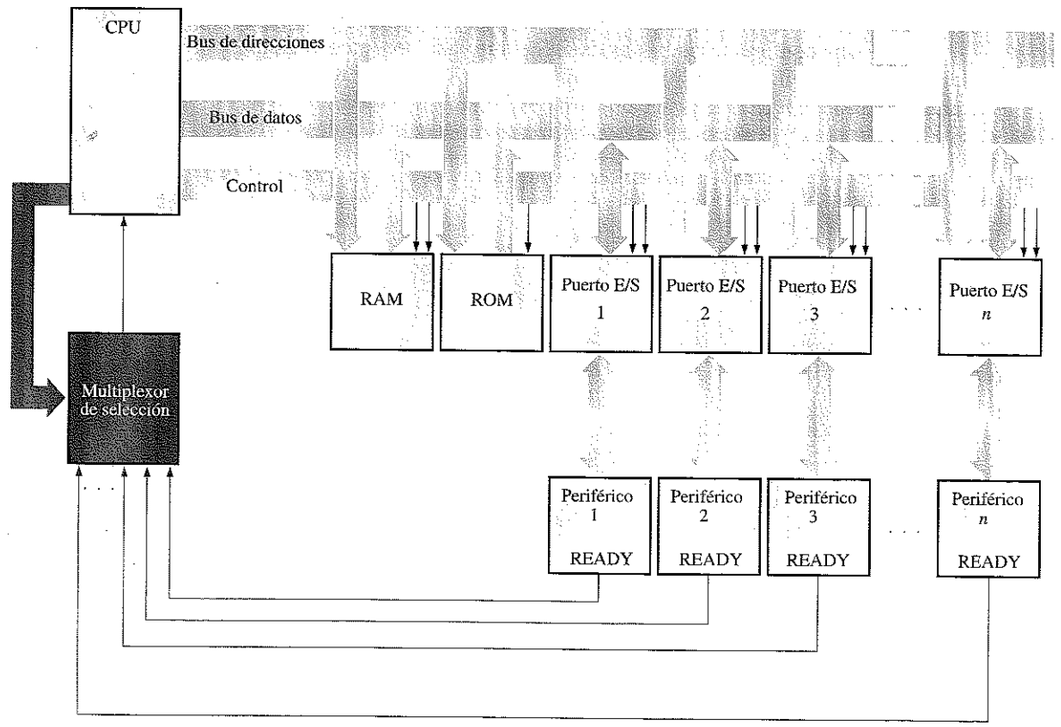


FIGURA 14.35
Configuración básica de la E/S por sondeo.

E/S controlada por interrupción

Este método supera las desventajas del método por sondeo. En el método por interrupción, la CPU responde a las necesidades de servicio sólo cuando el dispositivo periférico solicita el servicio. De este modo, la CPU se puede concentrar en la ejecución del programa en curso, sin tener que interrumpirlo innecesariamente para ver si los periféricos necesitan servicio.

Cuando la CPU recibe una señal de interrupción de E/S, detiene temporalmente el programa en curso, confirma la aceptación de la interrupción y carga de la memoria un programa especial (rutina de servicio) para el dispositivo particular que ha generado la interrupción. Cuando se completa la rutina de servicio, la CPU vuelve al punto donde estaba.

Un dispositivo denominado controlador programable de interrupciones (**PIC,** programmable interrupt controller) maneja las interrupciones de acuerdo con su prioridad, aceptando solicitudes de servicio procedentes de los periféricos. Si dos o más periféricos solicitan servicio al mismo tiempo, el que tenga asignada la prioridad más alta recibe el servicio en primer lugar, y luego el que tenga la siguiente prioridad más alta, y así sucesivamente. Después de enviar una señal de interrupción (*INTR*) a la CPU, el PIC proporciona a la CPU la dirección de comienzo en memoria de la rutina

de servicio apropiada. Este proceso se llama *vectorización*. La Figura 14.36 muestra una configuración básica de E/S controlada por interrupción.

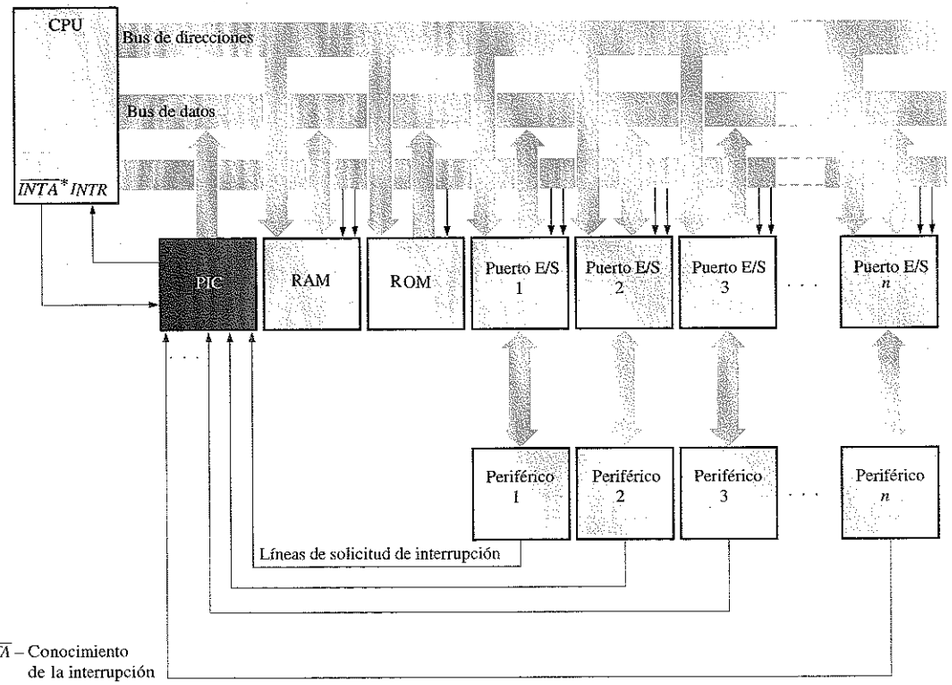


FIGURA 14.36
Configuración básica de E/S controlada por interrupción

Interrupciones software

Otro tipo de interrupción es la denominada **interrupción software**. Estas interrupciones son instrucciones de programa que pueden invocar a las rutinas de servicio descritas anteriormente. La diferencia se encuentra en que, en este caso, se invocan por software, en lugar de mediante hardware externo. Cuando se invocan, la rutina de servicio de interrupción se ejecuta exactamente del mismo modo que si se hubiera producido una interrupción hardware. Las primeras cinco interrupciones han sido definidas por Intel. Otras son definidas por el sistema BIOS y por el sistema operativo DOS para llevar a cabo muchas operaciones de E/S, como la lectura y escritura de datos en disco, la presentación de datos en pantalla y la lectura de datos desde el teclado.

REVISIÓN DE LA SECCIÓN 14.8

1. ¿En qué se diferencia una E/S por sondeo de una E/S controlada por interrupción?
2. ¿Cuál es la principal ventaja de la E/S controlada por interrupción?
3. ¿Qué es una interrupción software?

14.9 ■ ACCESO DIRECTO A MEMORIA (DMA)

En esta corta sección, se define la técnica de transferencia de datos denominada acceso directo a memoria (DMA, direct memory access), presentándose una comparación entre las transferencias realizadas por la CPU y las transferencias DMA.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Definir el término *DMA*. ■ Comparar una transferencia de datos de E/S de memoria realizada por la CPU y una transferencia DMA.

Todas las transferencias de datos de E/S estudiadas hasta el momento se han hecho a través de la CPU. Por ejemplo, cuando los datos se transfieren desde la RAM a un dispositivo periférico, la CPU lee el primer byte de datos de la memoria y lo carga en un registro interno del microprocesador.

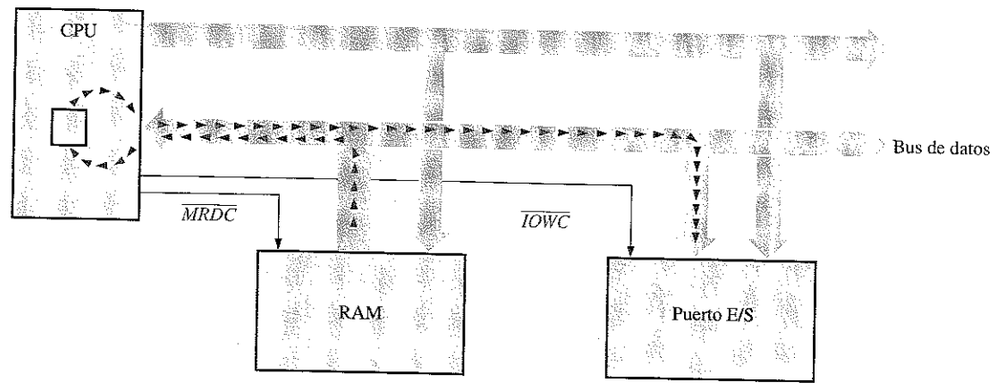


FIGURA 14.37
Transferencia de E/S de memoria realizada por la CPU.

Después, la CPU escribe el byte de datos en el correspondiente puerto de E/S. Esta operación de lectura/escritura se repite para cada byte del grupo de datos que haya que transferir. La Figura 14.37 ilustra este proceso.

Para bloques de datos largos, las paradas intermedias del microprocesador consumen mucho tiempo. Por esta razón, muchos sistemas utilizan la técnica denominada acceso directo a memoria (**DMA**) para acelerar las transferencias entre la RAM y determinados dispositivos periféricos. Básicamente, para ciertos tipos de transferencias de datos, el acceso directo a memoria ignora a la CPU, eliminando el tiempo consumido por los ciclos de carga y ejecución normales requeridos para cada operación de lectura o escritura.

Para la transferencia directa de memoria se utiliza un dispositivo denominado *controlador de DMA*, que toma el control de los buses del sistema y permite que los datos fluyan directamente entre la RAM y los periféricos, como muestra la Figura 14.38. En particular, las transferencias entre la unidad de disco y la RAM se hacen con

este método, debido a la gran cantidad de datos implicados y a la naturaleza secuencial de las transferencias. El controlador de DMA puede efectuar transferencias de datos varias veces más rápidamente que la CPU.

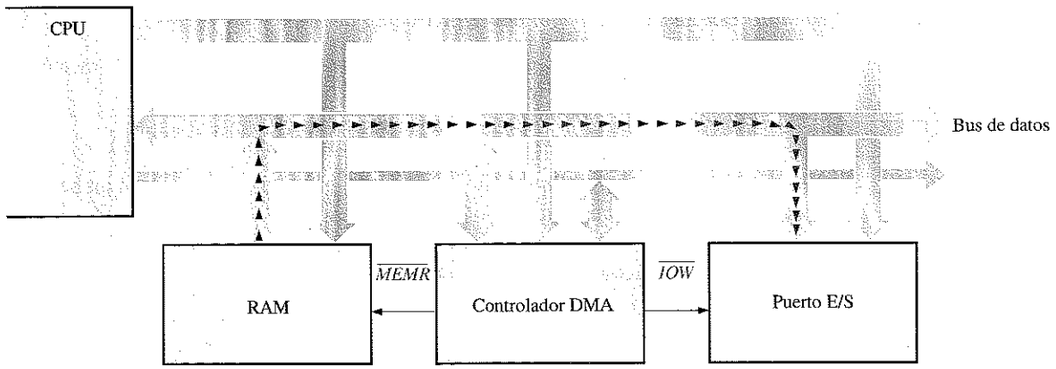


FIGURA 14.38
Transferencia por acceso directo a memoria.

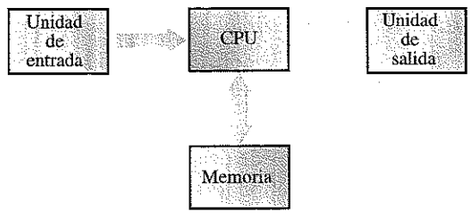
REVISIÓN DE LA SECCIÓN 14.9

1. ¿Qué significa DMA?
2. Comentar la ventaja del mecanismo DMA, y dar un ejemplo de un tipo de transferencia para el que se utiliza frecuentemente.

■ RESUMEN

- Las unidades básicas de una computadora se muestran en la Figura 14.39.

FIGURA 14.39



- Los tres buses básicos de una computadora son el *bus de direcciones*, el *bus de datos* y el *bus de control*. El tamaño de cualquiera de estos buses viene especificado por el número de hilos conductores distintos.
- Entre los dispositivos periféricos típicos se incluyen el teclado, las unidades de disco externas, el ratón, la impresora, el modem y el escáner.
- El número de líneas de dirección varía desde 20, en el 8086 /8088, hasta 32 para la familia de procesadores Pentium. Originalmente, el bus de datos era de 16 bits para el 8086 y ahora es de 64 para la familia Pentium.

- Los registros de propósito general del 8086/8088 son un subconjunto de los existentes en todos los procesadores 80X86 y Pentium de Intel. Dichos registros incluyen:
  - Acumulador (AX, que incluye AH y AL)
  - Base (BX, que incluye BH y BL)
  - Contador (CX, que incluye CH y CL)
  - Datos (DX, que incluye DH y DL)
  - Puntero de pila (SP)
  - Puntero de base (BP)
  - Índice de origen (SI)
  - Índice de destino (DI)

  A partir del 80386, este conjunto básico se amplió al conjunto de registros extendido.
- Los registros de segmento básicos para el 8086/8088 son un subconjunto de los registros existentes en todos los procesadores 80X86 y Pentium de Intel. Los registros de segmento son:
  - Segmento de código (CS)
  - Segmento de datos (DS)
  - Segmento de pila (SS)
  - Segmento extra (ES)

  A partir del 80386, se añadieron dos nuevos registros de segmento.
- Los registros indicadores (flag) para el 8086/8088 son un subconjunto de los registros existentes en todos los procesadores Intel. Los registros indicadores incluyen:
  - Captura (TF)
  - Dirección (DF)
  - Habilitación de interrupción (IF)
  - Desbordamiento (OF)
  - Acarreo (CF)
  - Signo (SF)
  - Cero (ZF)
  - Acarreo auxiliar (AF)
  - Paridad (PF)
- El lenguaje "básico" de una computadora se denomina código máquina y sus instrucciones se especifican como una serie de códigos binarios.
- En el lenguaje ensamblador, las instrucciones en código máquina se reemplazan por mnemónicos alfabéticos abreviados que tienen una correspondencia directa con el código máquina. El lenguaje ensamblador también utiliza directivas que permiten al programador especificar otros parámetros que no se traducen directamente a código máquina.
- Los puertos constituyen una interfaz con los dispositivos externos. Se pueden configurar como puertos de entrada, de salida o una combinación de ambos. Puede accederse a ellos como puertos dedicados o asignados a memoria y se les puede dar servicio mediante sondeo, mediante interrupciones o mediante software.

■ PALABRAS CLAVE

Estos términos también se encuentran en el glosario del final del libro.

Acumulador Registro de propósito general utilizado para las operaciones aritméticas y lógicas.

ALU (Arithmetic Logic Unit) Unidad aritmético-lógica; el elemento clave de procesamiento de un microprocesador, que realiza operaciones aritméticas y lógicas.

Arquitectura Disposición funcional interna de los elementos, que dota a un dispositivo de características de operación particulares.

BIOS Sistema básico de entrada/salida. Conjunto de programas en ROM que establece la interfaz con los dispositivos de E/S en un sistema de computadora.

BIU Unidad de interfaz de bus. La parte de la CPU que establece la interfaz con los buses del sistema y carga las instrucciones, lee los operandos y escribe los resultados.

Bus de control Un conjunto de conductores que conecta la CPU con otras partes de la computadora, para coordinar sus operaciones y comunicarse con dispositivos externos.

Bus de datos Un conjunto bidireccional de conductores por los que los datos o códigos de instrucciones se transfieren al microprocesador o por los que se envía, desde el microprocesador, el resultado de una operación o cálculo.

Bus de direcciones Generalmente, un grupo de conductores de un solo sentido que va desde el microprocesador a la memoria y que contiene la información de direcciones.

Cadena Una secuencia contigua de bytes o palabras.

Código de operación Código que representa una instrucción; un mnemónico.

Código máquina Instrucciones básicas en código binario que el procesador es capaz de comprender.

Código reubicable Programa que puede desplazarse a cualquier posición dentro del espacio de memoria sin tener que modificar el código básico.

Cola Memoria de alta velocidad que almacena instrucciones o datos.

Compilador Software que traduce instrucciones de programas de alto nivel, como BASIC, Pascal o Fortran, a lenguaje máquina.

Contiguo Combinado.

Coprocesador Microprocesador diseñado con un conjunto de instrucciones limitado que ha sido optimizado para realizar operaciones aritméticas muy rápidamente; generalmente, trabaja junto con un microprocesador de propósito general.

CPU Central Processing Unit, unidad central de proceso. Uno de los componentes principales de todas las computadoras, que controla el funcionamiento interno y procesa los datos.

Debug Programa incluido en el sistema operativo DOS, que permite realizar diversas operaciones sobre archivos. Incluye un programa ensamblador rudimentario.

Dirección Posición de memoria diferenciada, que contiene un byte.

Directiva de ensamblador Instrucción destinada al programa ensamblador. También se denomina pseudooperación.

DMA Direct Memory Access, acceso directo a memoria. Método para conectar de forma directa un dispositivo periférico con la memoria, sin utilizar la CPU como mecanismo de control.

Emparejamiento de instrucciones El proceso de combinar ciertas instrucciones independientes, de modo que puedan ser ejecutadas simultáneamente por dos unidades de ejecución separadas.

Entrada/salida asignada a memoria Un método para direccionar puertos, en el que se asigna a éstos direcciones pertenecientes al espacio de direcciones de memoria de la computadora. La CPU ve los puertos de E/S como posiciones de memoria.

Hardware La circuitería y componentes físicos de un sistema de computadora (en oposición a las instrucciones que denominamos software).

Interrupción Señal o instrucción que hace que el proceso actual sea temporalmente detenido mientras se ejecuta una rutina de servicio.

IP Puntero de instrucción. Un registro especial de la CPU que almacena el desplazamiento correspondiente a la siguiente instrucción que se va a ejecutar.

Lenguaje ensamblador Un tipo de lenguaje de programación de computadoras, en el que el código se expresa mediante mnemónicos que representan las instrucciones.

Matriz de registros Un conjunto de posiciones de almacenamiento temporal dentro del microprocesador, para almacenar datos y direcciones a las que tiene que acceder rápidamente el programa.

Microcontrolador Microprocesador especializado diseñado para funciones de control.

Microprocesador Circuito integrado digital que puede ser programado mediante una serie de instrucciones para realizar funciones específicas sobre los datos.

Mnemónico Instrucción similar al idioma inglés que es convertida por un programa ensamblador en código máquina para que lo use un procesador.

Modo real Modo de operación de un procesador Intel que emula la memoria de 1 Mbyte del 8086.

Multitarea Entorno de sistema operativo en el que la computadora parece ejecutar múltiples programas o tareas simultáneamente.

Operando Variable, registro, posición de memoria o valor utilizado como parte de la instrucción en programas escritos en lenguaje ensamblador.

PIC Controlador de interrupciones programable; maneja las instrucciones de acuerdo con su prioridad.

Pipeline Véase Precarga.

PPI Interfaz periférica programable. Circuito integrado especial utilizado para implementar puertos de E/S.

Precarga El proceso de ejecutar instrucciones al mismo tiempo que otras instrucciones son "cargadas", eliminando tiempo de inactividad. Este proceso también se denomina *pipeline.*

Programa Grupo de instrucciones creado para resolver un problema específico.

Programa ensamblador Programa que convierte los mnemónicos en código máquina.

Pseudo-operación Instrucción dirigida al programa ensamblador (en oposición a las instrucciones dirigidas a un procesador).

Puerto Una interfaz física en una computadora, a través de la cual se transfieren datos hacia o desde un periférico.

Puerto de E/S Puerto de entrada/salida. La interfaz entre un bus interno y un periférico.

Puntero El contenido de un registro que almacena una dirección.

Segmento Un bloque de memoria de 64k.

Software Programas e instrucciones.

Sondeo El proceso de comprobar una serie de dispositivos periféricos para determinar si alguno de ellos requiere servicio por parte de la CPU.

Subrutina Serie de instrucciones que se pueden agrupar y que un programa puede usar de forma repetida, pero que sólo se escribe una vez.

Tasa de procesamiento La velocidad media con la que se ejecuta un programa.

Unidad de control La parte del microprocesador que proporciona las señales de temporización y control para introducir y extraer datos del microprocesador y para sincronizar la ejecución de instrucciones.

Unidad de ejecución (EU) La parte de la CPU que ejecuta las instrucciones. Contiene la unidad aritmético-lógica (ALU), los registros de propósito general y los indicadores (*flag*).

■ AUTOTEST

1. Una computadora básica no incluye
   (a) unidad aritmético-lógica (b) unidad de control
   (c) unidades periféricas (d) unidad de memoria
2. Un bus de direcciones de 20 bits admite
   (a) 100.000 direcciones de memoria (b) 1.048.576 direcciones de memoria
   (c) 2.097.152 direcciones de memoria (d) 20.000 direcciones de memoria
3. El tamaño del bus de datos del procesador Pentium es
   (a) 16 (b) 24 (c) 32 (d) 64
4. Un bus que se usa para transferir información desde y hacia el microprocesador es el
   (a) bus de direcciones (b) bus de datos
   (c) los dos anteriores (d) ninguno de los anteriores
5 Un ejemplo de unidad periférica es
   (a) un registro de dirección (b) la MPU
   (c) el monitor de vídeo (d) el adaptador de interfaz
6. Los dos tipos de transferencias de memoria que emplea la CPU son
   (a) directa y por interrupción (b) lectura y escritura
   (c) por bus y multiplexada (d) entrada y salida
7. Dos tipos de puertos de E/S son
   (a) dedicado y asignado a memoria (b) DMA y PPI
   (c) de tres estados y octal (d) lectura y escritura
8. En la familia 80X86 de Intel, el número máximo de dispositivos de E/S de 8 bits es
   (a) 64 (b) 1000 (c) 64.000 (d) 1 millón (e) ilimitado

9. El sondeo es un método utilizado para
   (a) determinar el estado del microprocesador
   (b) establecer la comunicación entre la CPU y un periférico
   (c) establecer una prioridad para la comunicación con varios periféricos
   (d) determinar la siguiente instrucción
10. De los siguientes registros, ¿cuál es un registro de 8 bits?
    (a) AH (b) BX (c) SS (d) IP
11. Esencialmente, un mnemónico es
    (a) un organigrama (b) un operando
    (c) código máquina (d) una instrucción
12. DMA quiere decir
    (a) digital microprocessor address (dirección de microprocesador digital)
    (b) direct memory access (acceso directo a memoria)
    (c) data multiplexed access (acceso multiplexado a datos)
    (d) direct memory addressing (direccionamiento directo de memoria)
13. Un programa de computadora es una lista de
    (a) direcciones de memoria que contienen los datos que hay que usar en una operación
    (b) direcciones que contienen instrucciones que hay que utilizar en una operación
    (c) instrucciones ordenadas para conseguir un resultado específico
14. Un tipo de estructura en lenguaje ensamblador que altera el curso del programa se denomina
    (a) bucle (b) salto
    (c) los dos anteriores (d) ninguno de los anteriores
15. Un tipo de interrupción que se invoca desde dentro de un programa se denomina
    (a) interrupción software (b) interrupción por sondeo
    (c) interrupción directa (d) interrupción de E/S

■ CUESTIONES

SECCIÓN 14.1 El microprocesador y la computadora

1. Nombrar los tres elementos básicos de un microprocesador.
2. (a) ¿Qué es un coprocesador?
   (b) ¿Dónde se encuentra el coprocesador en el Pentium?
3. Nombrar los tres buses básicos de una computadora y explicar para qué se usa cada uno.
4. ¿Qué es un puerto?

SECCIÓN 14.2 Familias de microprocesadores

5. (a) Nombrar el primer microprocesador de Intel que utilizó un bus de direcciones de 32 bits.
   (b) ¿Cuántas posiciones podía direccionar?
6. ¿Cuántas líneas de datos se usan en el Pentium?
7. Nombrar el primer microprocesador de Motorola que empleó un bus de direcciones de 32 bits.
8. ¿Cuál es la principal diferencia entre las series de procesadores MPC601 y 68000 de Motorola?

SECCIÓN 14.3 El microprocesador 8086/8088 y el modelo de software del procesador Pentium

9. ¿Cuáles son los tres pasos básicos que repite de forma cíclica un procesador?
10. ¿Qué significa "pipeline"?
11. Nombrar los seis registros de segmento del 80386 y de los procesadores siguientes.

12. Suponer que el registro de segmento de código contiene el número hexadecimal 0F05 y que el registro puntero de instrucción contiene el número 0100. ¿Cuál es la dirección física de la siguiente instrucción que se va a ejecutar?
13. Explicar la diferencia entre los registros AH, AL, AX y EAX.
14. **(a)** ¿Qué es un indicador (*flag*)?
    (b) ¿Para qué dos propósitos se usan los indicadores?
15. **(a)** Explicar la ventaja del emparejamiento de instrucciones en el procesador Pentium.
    (b) ¿Por qué no se puede emplear el emparejamiento de instrucciones en el 80486?

SECCIÓN 14.4 Programación del microprocesador

16. ¿Qué es un programa ensamblador?
17. Dibujar un organigrama para un programa que sume todos los números desde 1 hasta 10 y almacene el resultado en una posición de memoria denominada TOTAL.
18. Dibujar un organigrama que muestre cómo se podría contar el número de bytes de una cadena y colocar el resultado en una posición de memoria denominada COUNT. Suponer que la cadena comienza en una posición denominada INICIO y tiene un carácter 20 hexadecimal (equivalente ASCII del carácter espacio) para indicar el final de la misma. El carácter de espacio no debe contarse.
19. ¿Qué hace la directiva de ensamblador **word ptr**?
20. Explicar qué ocurre cuando se ejecuta la instrucción **mov ax,[bx]**
21. Describir los siete tipos básicos de instrucciones para los procesadores Intel.

SECCIÓN 14.5 La unidad central de procesamiento (CPU)

22. ¿Cuál es la diferencia entre las señales $\overline{IORC}$ y $\overline{MRDC}$?
23. ¿Qué líneas de dirección se usan en las operaciones de lectura y escritura de E/S en el 8086/8088?
24. ¿Cuál es el propósito de la señal *ALE* del controlador de bus del 8086/8088?
25. ¿Cuál es la función de los bits de estado de la CPU 8086/8088?
26. Nombrar al menos tres mejoras del procesador Pentium que no existían en el microprocesador original 8086/8088.

SECCIÓN 14.6 La memoria

27. ¿Qué posiciones de memoria se reservan para el área de trabajo del BIOS?
28. ¿Qué significa el término *memoria extendida*?
29. Cuando el controlador de bus envía el código de estado $\overline{S_2}\,\overline{S_1}\,\overline{S_0} = 110$, ¿qué tipo de operación indica?

SECCIÓN 14.7 El puerto de entrada/salida (E/S)

30. Explicar la diferencia entre puerto de E/S dedicado y mapeado en memoria. ¿Cuál de ellos no puede emplear las instrucciones IN y OUT?
31. ¿Cuál es el número máximo de puertos dedicados que pueden conectarse a un procesador Pentium?
32. Explicar la función de un circuito integrado 8255A.

SECCIÓN 14.8 Interrupciones

33. Comparar la E/S por sondeo y la E/S controlada por interrupción.
34. ¿Qué significa el término *vectorización*?
35. ¿Qué es una interrupción software?

SECCIÓN 14.9 Acceso directo a memoria (DMA)

36. Explicar qué ocurre en una operación de acceso directo a memoria.

■ RESPUESTAS A LAS REVISIONES DE CADA SECCIÓN

SECCIÓN 14.1

1. Una computadora incluye una unidad central de procesamiento (CPU), memoria y puertos de entrada/salida.
2. Los tres buses son: (1) el bus de datos que transfiere los datos a y desde la CPU, (2) el bus de direcciones que lleva las direcciones de memoria o de puerto, y (3) el bus de control que transporta las señales de temporización y control.
3. Datos e instrucciones.
4. El lenguaje ensamblador es un lenguaje de bajo nivel que puede traducirse de forma directa a instrucciones binarias o patrones de datos. Los dos tipos de operaciones son las instrucciones y las pseudo-operaciones (directivas).

SECCIÓN 14.2

1. Un microprocesador de 32 bits tiene un bus de datos de 32 bits.
2. El 8086 fue el primer microprocesador de 16 bits de Intel.
3. RISC quiere decir Reduced Instruction Set Computer, computadora con conjunto de instrucciones reducido.
4. La memoria caché almacena instrucciones o datos utilizados recientemente, para acelerar el funcionamiento.
5. Superescalar hace referencia a un tipo de arquitectura que permite que se ejecute más de una instrucción durante un ciclo de reloj.

SECCIÓN 14.3

1. Los registros de propósito general del 8086/8088 son:

Acumulador (AX: AH, AL)	Puntero de pila (SP)
Índice base (BX:BH, BL)	Puntero de base (BP)
Contador (CX:CH, CL)	Índice de destino (DI)
Datos (DX:DH. DL)	Índice de origen (SI)

2. La BIU proporciona una interfaz para los datos y direcciones.
3. No, la unidad de ejecución no actúa de interfaz con los buses.
4. La cola de instrucciones almacena instrucciones precargadas para la unidad de ejecución, con el fin de aumentar la tasa de procesamiento.
5. Los programas se pueden reubicar fácilmente dentro de la memoria.
6. El emparejamiento de instrucciones es el proceso de combinar instrucciones independientes, de modo que puedan ejecutarse simultáneamente mediante las dos unidades de ejecución del Pentium.

SECCIÓN 14.4

1. Un programa es una lista de instrucciones de computadora ordenadas para conseguir un resultado específico.
2. Un conjunto complejo de instrucciones dispone de un completo repertorio de instrucciones, proporcionando al programador una gran flexibilidad.
3. Un código de operación es el código de una instrucción.
4. Un operando es la cantidad o elemento con el que se opera en una instrucción.
5. Una cadena es una secuencia contigua de bytes o palabras.

SECCIÓN 14.5

1. El generador de reloj proporciona las señales de temporización básicas al microprocesador.
2. El controlador de bus ejecuta comandos de lectura y escritura de memoria, ejecuta comandos de lectura y escritura de E/S y habilita los latches de dirección y el transmisor/receptor del bus de datos.
3. En el 8088, las veinte líneas de bus disponibles deben utilizarse para el bus de direcciones de 20 bits y el bus de datos de 8 bits.
4. La técnica de pipeline, que permite ejecutar algunas instrucciones en un solo ciclo de reloj; la adición y ampliación de la memoria caché; la inclusión del coprocesador matemático interno; los procesadores dobles internos para operaciones con enteros y la lógica de predicción de saltos.

SECCIÓN 14.6

1. Los dos ciclos de la CPU son lectura y escritura.
2. Cuatro ciclos de reloj.
3. Una dirección válida aparece en el bus después del flanco anterior del pulso *ALE*.

SECCIÓN 14.7

1. Un puerto de E/S es la interfaz entre el bus de datos interno y un periférico.
2. Un puerto de E/S dedicado tiene una dirección asignada dentro del espacio de direcciones de E/S. Un puerto de E/S asignado a memoria tiene una dirección dentro del espacio de direcciones de memoria.
3. Una PPI proporciona puertos de E/S.

SECCIÓN 14.8

1. En la E/S controlada por interrupción, la CPU proporciona servicio a los periféricos sólo cuando el periférico lo solicita. En la E/S por sondeo, la CPU comprueba periódicamente el periférico para ver si necesita servicio.
2. Las E/S controladas por interrupción ahorran tiempo de CPU.
3. Una interrupción software es una instrucción que invoca una rutina de servicio de interrupción.

SECCIÓN 14.9

1. DMA corresponde a direct memory access, acceso directo a memoria.
2. Una transferencia DMA de datos de la memoria a la E/S o viceversa ahorra tiempo de CPU. A menudo, el acceso directo a memoria se usa para transferir datos entre la RAM y una unidad de disco.

■ RESPUESTAS A LOS PROBLEMAS RELACIONADOS DE LOS EJEMPLOS

1. $6C4C2_{16}$
2. Cambiar el primer bloque (bloque de inicialización) a "BIG = FFFF"; se trata del número sin signo mayor posible. Cambiar la primera pregunta por "Is number <BIG?"

■ RESPUESTAS AL AUTOTEST

1. (c) **2.** (b) **3.** (d) **4.** (b) **5.** (c) **6.** (b) **7.** (a) **8.** (c)
9. (b) **10.** (a) **11.** (d) **12.** (b) **13.** (c) **14.** (c) **15.** (a)

15

TECNOLOGÍAS DE LOS CIRCUITOS INTEGRADOS

15.1 Parámetros y características de operación básicas
15.2 Circuitos CMOS
15.3 Circuitos TTL
15.4 Consideraciones prácticas sobre el uso de TTL
15.5 Comparación de las prestaciones CMOS y TTL
15.6 Circuitos ECL
15.7 PMOS, NMOS y E²CMOS

WWW Simulación por computadora de circuitos digitales:
El tutorial de Electronics Workbench (EWB) se encuentra en la dirección http://www.prenhall.com/floyd.

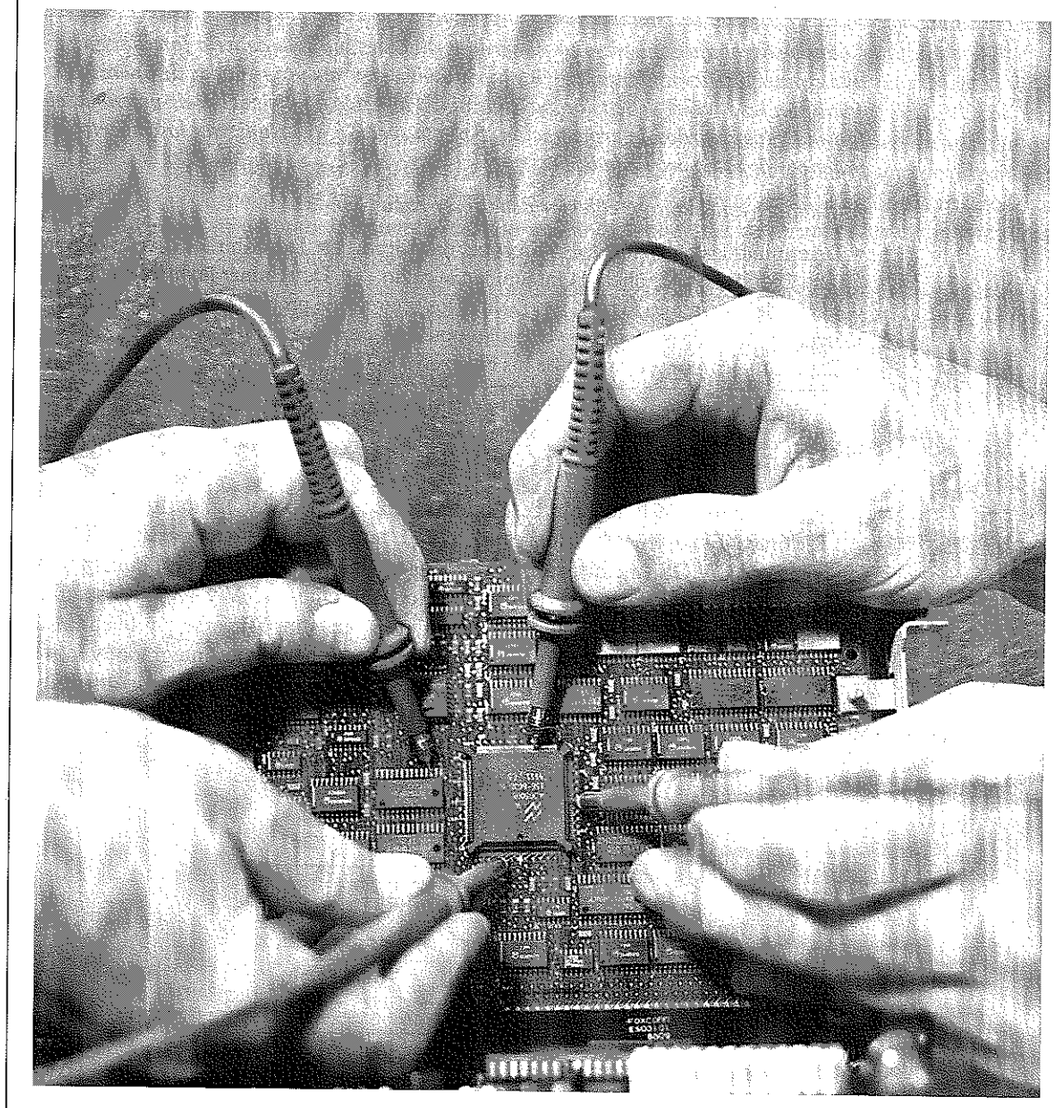
■ OBJETIVOS DEL CAPÍTULO

- ❑ Determinar el margen de ruido de un dispositivo a partir de los parámetros de la hoja de características.
- ❑ Calcular la disipación de potencia de un dispositivo.
- ❑ Explicar cómo afecta el retardo de propagación a la frecuencia de funcionamiento o velocidad de un circuito.
- ❑ Interpretar el producto velocidad-potencia como una medida de rendimiento.
- ❑ Utilizar las hojas de características para obtener información sobre un dispositivo específico.
- ❑ Explicar qué es el fan-out de una puerta.
- ❑ Describir cómo funcionan las puertas TTL y CMOS básicas en el nivel de componentes.
- ❑ Establecer la diferencia entre salidas TTL totem-pole y salidas TTL en colector abierto, y entender las limitaciones y usos de cada una de ellas.
- ❑ Conectar circuitos en configuración AND-cableada.
- ❑ Describir el funcionamiento de los circuitos triestado.
- ❑ Terminar adecuadamente las entradas de las puertas no utilizadas.
- ❑ Comparar las características de las familias TTL y CMOS.
- ❑ Manipular los dispositivos CMOS sin riesgos de daños debidos a las descargas electrostáticas.
- ❑ Indicar las ventajas de la tecnología ECL.
- ❑ Describir los circuitos PMOS y NMOS.
- ❑ Describir una celda E²CMOS.

■ INTRODUCCIÓN

La intención de este capítulo es que se utilice como capítulo "flotante". Puede estudiarse en diferentes puntos del libro u omitirse por completo, dependiendo de los objetivos del curso y de las preferencias personales. Como sugerencia, los puntos que nos parecen más adecuados para abordarlo son después de los capítulos 3 o 6, pero, si se desea, puede verse después de cualquier otro capítulo. También, determinados temas de este capítulo pueden introducirse en diversos puntos del libro. En cualquier caso, la Sección 3.7 debería estudiarse antes de comenzar con este capítulo.

Finalmente, este capítulo puede también concebirse como lectura opcional o utilizarse como referencia. Obsérvese que se ha incluido, en el margen derecho de ciertas páginas, un diseño que facilita el hacer referencia a este capítulo.



15.1 ■ PARÁMETROS Y CARACTERÍSTICAS DE OPERACIÓN BÁSICAS

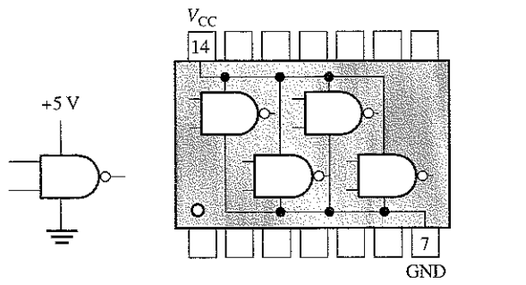
Cuando se trabaja con circuitos integrados digitales, no sólo debería familiarizarse uno con su funcionamiento lógico, sino también con sus propiedades de operación, como son los niveles de tensión, la inmunidad al ruido, la disipación de potencia, el fan-out y los retardos de propagación. En esta sección, se estudiarán los aspectos prácticos de estas propiedades.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Determinar las conexiones de alimentación y tierra. ■ Describir los niveles lógicos TTL y CMOS. ■ Explicar la inmunidad al ruido. ■ Determinar la disipación de potencia de un circuito lógico. ■ Definir los retardos de propagación de una puerta lógica. ■ Definir el producto velocidad-potencia y explicar su importancia. ■ Explicar la carga y el fan-out de los dispositivos TTL y CMOS.

Tensión de alimentación continua

El valor nominal de la tensión de alimentación continua (DC) para los dispositivos **TTL** (lógica transistor-transistor) es +5 V. TTL también se designa mediante T^2L. Los dispositivos **CMOS** (metal óxido semiconductor complementario) están disponibles en dos categorías de tensiones de alimentación diferentes, +5 V y +3,3 V; ésta última se conoce como CMOS de baja tensión. Aunque para simplificar se omite en los diagramas lógicos, esta tensión se conecta al pin V_{CC} de un circuito integrado, y la masa se conecta al pin de masa (GND). Tanto la alimentación como la masa se distribuyen internamente a todos los elementos del chip, como se ilustra en la Figura 15.1 para un encapsulado de 14 pines.



(a) Una única puerta (b) Circuito integrado en encapsulado DIL

FIGURA 15.1
Ejemplo de las conexiones y distribución de V_{CC} y masa en un circuito integrado. Para simplificar, el resto de las conexiones se han omitido.

Niveles lógicos CMOS

En el Capítulo 1 se estudiaron brevemente los niveles lógicos. Existen cuatro especificaciones diferentes para los niveles lógicos: V_{IL}, V_{IH}, V_{OL} y V_{OH}. Para los circuitos CMOS, el rango de las tensiones de entrada (V_{IL}) que representan un nivel lógico

BAJO (0 lógico) válido va de 0 V a 1,5 V para la lógica de +5 V y de 0 V a 0,8 V para la lógica de 3,3 V. El rango de las tensiones de entrada (V_{IH}) que pueden representar un nivel ALTO (1 lógico) va de 3,5 a 5 V para la lógica de 5 V y de 2 V a 3,3 V para la lógica de 3,3 V, como se indica en la figura 15.2. El rango de valores entre 1,5 V y 3,5 V para la lógica de 5 V y el rango de 0,8 V a 2 V para la lógica de 3,3 V son regiones de funcionamiento no predecible, y los valores comprendidos en dichos rangos no están permitidos. Cuando una tensión de entrada se encuentra en uno de estos rangos, puede ser interpretada como un nivel ALTO o un nivel BAJO por el circuito lógico. Por tanto, las **puertas** CMOS no pueden funcionar de forma fiable cuando las tensiones se encuentran dentro de uno de estos rangos.

Los rangos de las tensiones de salida CMOS (V_{OL} y V_{OH}) para las lógicas de 5 V y 3,3 V se muestran en la Figura 15.2. Obsérvese que la tensión de salida mínima a nivel ALTO, $V_{OH(mín)}$, es mayor que la tensión de entrada mínima a nivel ALTO, $V_{IH(mín)}$; y que la tensión de salida máxima a nivel BAJO, $V_{OL(máx)}$, es menor que la tensión de entrada máxima a nivel BAJO, $V_{IL(máx)}$.

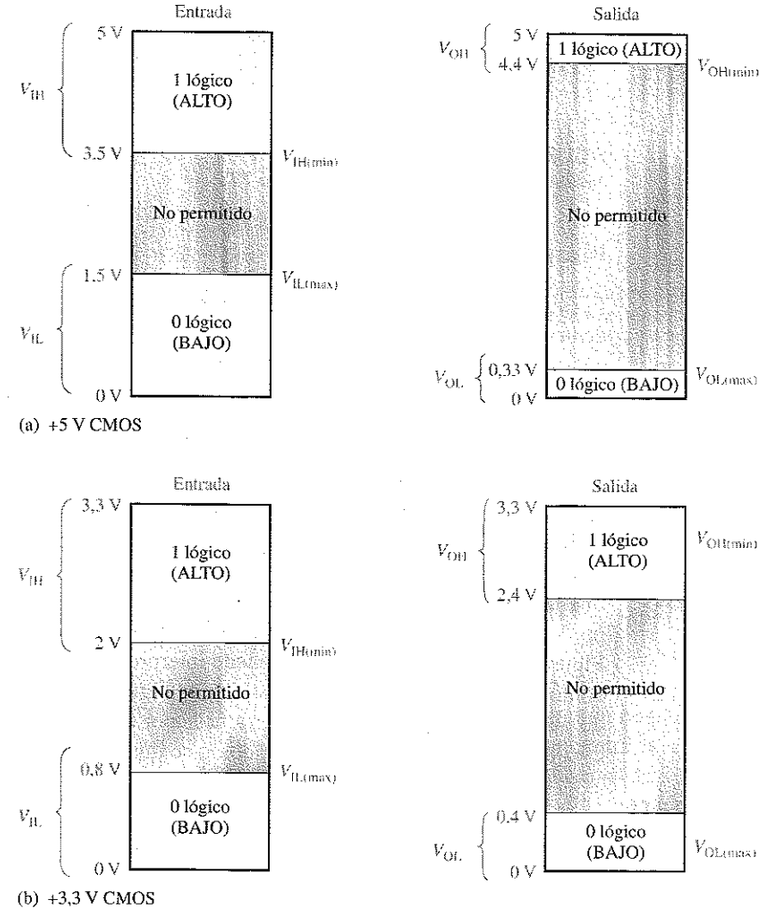


FIGURA 15.2
Niveles lógicos de entrada y de salida para los dispositivos CMOS.

Niveles lógicos TTL

En la Figura 15.3 se proporcionan los niveles lógicos de entrada y de salida para los dispositivos TTL. Al igual que para los dispositivos CMOS, existen cuatro especificaciones diferentes para los niveles lógicos: V_{IL}, V_{IH}, V_{OL} y V_{OH}.

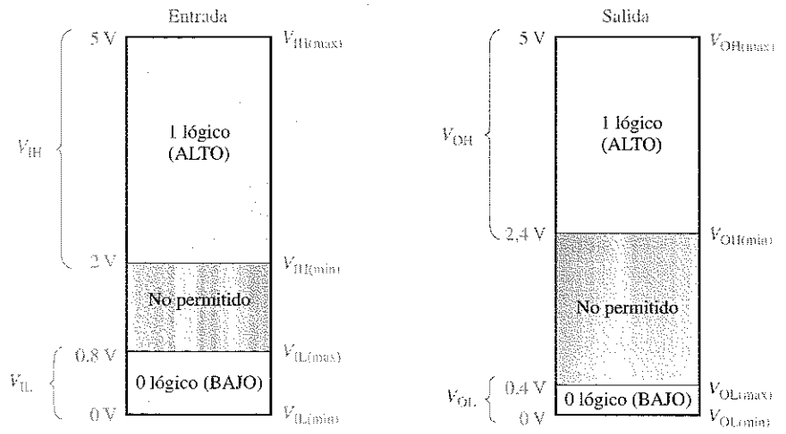


FIGURA 15.3
Niveles lógicos de entrada y salida para TTL.

Inmunidad al ruido

El ruido es una tensión no deseada que se induce en los circuitos eléctricos y que puede ser una amenaza para el correcto funcionamiento del circuito. Los cables y otros conductores internos del sistema pueden captar las radiaciones electromagnéticas de alta frecuencia de los conductores adyacentes, en los que las corrientes varían rápidamente, o de otras fuentes externas al sistema. También las fluctuaciones de tensión de la línea de alimentación son una forma de ruido de baja frecuencia.

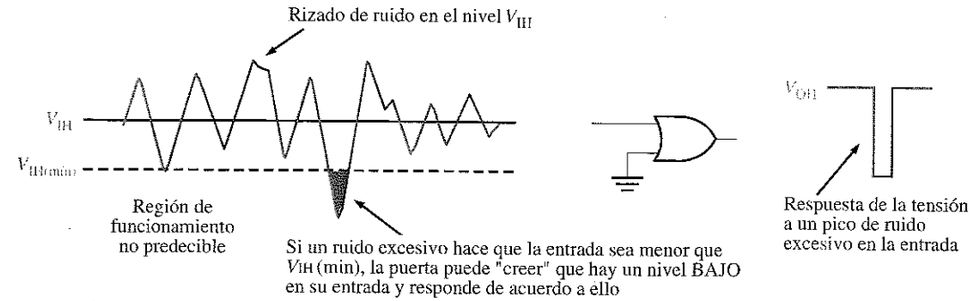
Para no verse afectados adversamente por el ruido, los circuitos lógicos deben tener cierta **inmunidad al ruido**, que es la capacidad de tolerar ciertas fluctuaciones de tensión no deseadas en sus entradas sin que cambie el estado de salida. Por ejemplo, si la tensión de ruido en la entrada de una puerta CMOS con lógica de +5 V hace que la tensión del nivel ALTO caiga por debajo de 3,5 V, el funcionamiento no será predecible, puesto que se encuentra en la región de operación no permitida (véase la Figura 15.2). Por tanto, la puerta puede interpretar la fluctuación por debajo de 3,5 V como un nivel BAJO, como se ilustra en la Figura 15.4(a). De forma similar, si el ruido hace que la entrada de una puerta pase por encima de 1,5 V en el nivel BAJO, se crea una condición indeterminada, como se ilustra en la parte (b).

Margen de ruido

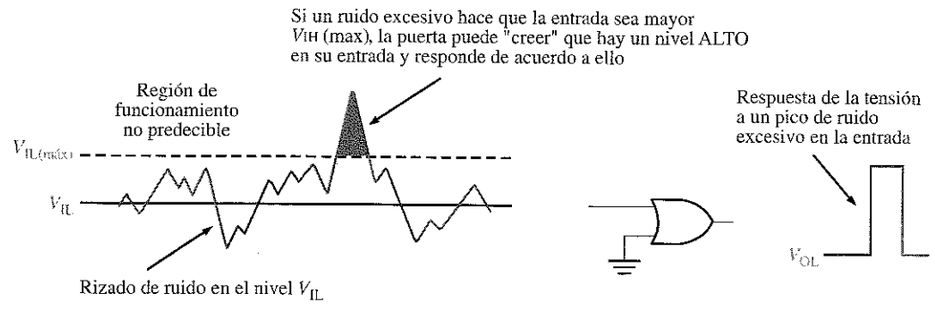
La medida de la inmunidad al ruido de un circuito se denomina **margen de ruido**, y se expresa en voltios. Para un determinado circuito, se especifican dos valores de margen de ruido: margen de ruido para el nivel ALTO (V_{NH}) y margen de ruido para el nivel BAJO (V_{NL}). Estos parámetros se definen mediante las siguientes ecuaciones:

$$V_{NH} = V_{OH(mín)} - V_{IH(mín)} \quad \textbf{(15.1)}$$

$$V_{NL} = V_{IL(máx)} - V_{OL(máx)} \quad \textbf{(15.2)}$$



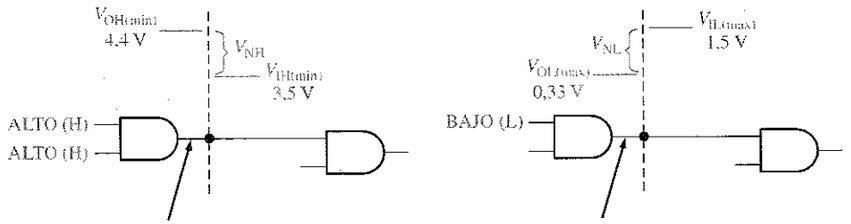
(a)



(b)

FIGURA 15.4
Efectos del ruido de entrada en el funcionamiento de la puerta.

En ocasiones, se verá que el margen de ruido se expresa como un porcentaje de V_{CC}. A partir de la ecuación, puede verse que V_{NH} es la diferencia entre la salida a nivel ALTO menor posible de una puerta excitadora ($V_{OH(mín)}$) y la entrada a nivel ALTO menor posible que la puerta de carga puede tolerar ($V_{IH(mín)}$). El margen de ruido V_{NL} es la diferencia entre la entrada a nivel BAJO máxima posible que la puerta puede tolerar ($V_{IL(máx)}$) y la salida a nivel bajo máxima posible de la puerta excitadora ($V_{OL(máx)}$). Los márgenes de ruido se ilustran en la Figura 15.5.



(a) Margen de ruido para el nivel ALTO

(b) Margen de ruido para el nivel BAJO

FIGURA 15.5
Márgenes de ruido. Los valores indicados son para dispositivos CMOS con lógica de 5 V, pero el principio se aplica a cualquier familia lógica.

EJEMPLO 15.1 Determinar los márgenes de ruido de los niveles ALTO y BAJO para TTL y CMOS, utilizando la información de las Figuras 15.2 y 15.3.

Solución. Para CMOS de 5 V,

$$V_{IH(mín)} = 3,5 \text{ V}$$
$$V_{IL(máx)} = 1,5 \text{ V}$$
$$V_{OH(mín)} = 4,4 \text{ V}$$
$$V_{OL(máx)} = 0,33 \text{ V}$$
$$V_{NH} = V_{OH(mín)} - V_{IH(mín)} = 4,4 \text{ V} - 3,5 \text{ V} = \mathbf{0,9 \text{ V}}$$
$$V_{NL} = V_{IL(máx)} - V_{OL(máx)} = 1,5 \text{ V} - 0,33 \text{ V} = \mathbf{1,17 \text{ V}}$$

Para TTL

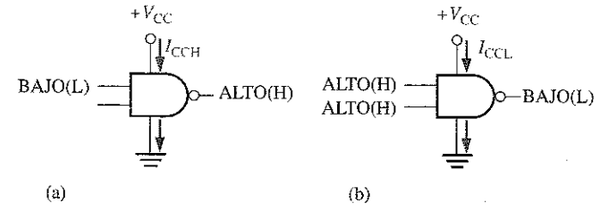
$$V_{IH(mín)} = 2 \text{ V}$$
$$V_{IL(máx)} = 0,8 \text{ V}$$
$$V_{OH(mín)} = 2,4 \text{ V}$$
$$V_{OL(máx)} = 0,4 \text{ V}$$
$$V_{NH} = V_{OH(mín)} - V_{IH(mín)} = 2,4 \text{ V} - 2 \text{ V} = \mathbf{0,4 \text{ V}}$$
$$V_{NL} = V_{IL(máx)} - V_{OL(máx)} = 0,8 \text{ V} - 0,4 \text{ V} = \mathbf{0,4 \text{ V}}$$

Una puerta TTL tiene una inmunidad de hasta 0,4 V de ruido para ambos estados de entrada, ALTO y BAJO.

Problema relacionado. Basándose en los cálculos anteriores de margen de ruido, ¿qué familia de dispositivos, CMOS de 5 V o TTL, debería utilizarse en entornos de alto ruido?

Disipación de potencia

Como se indica en la Figura 15.6, por una puerta lógica circula corriente procedente de una fuente de alimentación continua. Cuando el estado de salida de la puerta es un nivel ALTO, circula la corriente I_{CCH}, y cuando el estado de salida es un nivel BAJO, circula la corriente I_{CCL}.



(a) (b)

FIGURA 15.6
Corrientes de la fuente de alimentación DC.

Veamos un ejemplo. Si se especifica una I_{CCH} de 1,5 mA cuando V_{CC} es 5 V, y si la puerta está en un estado de salida estático (no cambia) ALTO, la **disipación de potencia** (P_D) de la puerta es

$$P_D = V_{CC}I_{CCH} = (5 \text{ V})(1,5 \text{ mA}) = 7,5 \text{ mW}$$

Cuando se aplican impulsos a la puerta, su salida conmuta entre los estados ALTO y BAJO, por lo que la corriente de alimentación varía entre I_{CCH} e I_{CCL}. La disipación de potencia media depende del ciclo de trabajo y, usualmente, se especifica para un ciclo de trabajo del 50%. Cuando el ciclo de trabajo es el 50%, la salida está a nivel ALTO la mitad del tiempo, y la mitad restante está a nivel BAJO. Por tanto, la corriente de alimentación media es:

$$I_{CC} = \frac{I_{CCH} + I_{CCL}}{2} \tag{15.3}$$

La disipación de potencia media es

$$P_D = V_{CC}I_{CC} \tag{15.4}$$

EJEMPLO 15.2 Por una determinada puerta circulan 2 μA cuando su salida está a nivel ALTO, y 3,6 μA cuando está a nivel BAJO. ¿Cuál es la disipación de potencia media si V_{CC} es 5 V y la puerta funciona con un ciclo de trabajo del 50%?

Solución. La corriente I_{CC} media es

$$I_{CC} = \frac{I_{CCH} + I_{CCL}}{2} = \frac{2,0\mu\text{A} + 3,6\mu\text{A}}{2} = 2,8\mu\text{A}$$

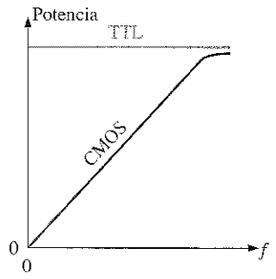
La disipación de potencia media es

$$P_D = V_{CC}I_{CC} = (5\text{ V})(2{,}8\ \mu\text{A}) = \mathbf{14\ \mu W}$$

Problema relacionado. Una cierta puerta de un circuito integrado tiene una I_{CCH} = 1,5 μA e I_{CCL} = 2,8 μA. Determinar la disipación media para un ciclo de trabajo del 50% si V_{CC} es 5V.

La disipación de potencia en un circuito TTL es esencialmente constante dentro de su rango de frecuencias de operación. Sin embargo, la disipación de potencia en CMOS depende de la frecuencia. En condiciones de estática (DC) es extremadamente baja y aumenta cuando aumenta la frecuencia. En las curvas generales de la Figura 15.7 se muestran estas características. Por ejemplo, la disipación de potencia de una puerta TTL Schottky de bajo consumo es 2,2 mW siempre (constante). La disipación de potencia de una puerta HCMOS es 2,75 μW bajo condiciones estáticas y 170 μW a 100 kHz.

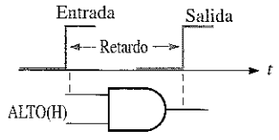
FIGURA 15.7
Curvas de potencia en función de la frecuencia para TTL y CMOS.



Retardo de propagación

Como ilustra la Figura 15.8, cuando una señal pasa (se propaga) a través de un circuito lógico, siempre experimenta un retardo temporal. Un cambio del nivel de salida siempre se produce en un cierto tiempo, llamado **retardo de propagación**, después de que se ha realizado el cambio de nivel en la entrada.

FIGURA 15.8
Ilustración básica del retardo de propagación.

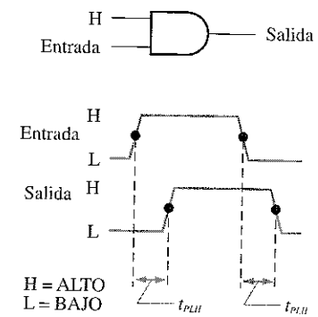


Como se mencionó en el Capítulo 3, existen dos retardos de propagación específicos para las puertas lógicas:

- t_{PHL}: es el tiempo entre un determinado punto del impulso de entrada y el correspondiente punto del impulso de salida, cuando la salida cambia de nivel ALTO a nivel BAJO.
- t_{PLH}: es el tiempo entre un determinado punto del impulso de entrada y el correspondiente punto del impulso de salida, cuando la salida cambia de nivel BAJO a nivel ALTO.

En la Figura 15.9 se ilustran estos retardos de propagación, utilizando como referencias los puntos del 50% de los flancos de los impulsos.

FIGURA 15.9
Retardos de propagación.



El retardo de propagación de una puerta limita la frecuencia a la que puede trabajar. Cuanto mayor es el retardo de propagación, menor es la frecuencia máxima. Luego un circuito de mayor velocidad tendrá un retardo de propagación menor. Por ejemplo, una puerta con un retardo de 3 ns es más rápida que otra con un retardo de 10 ns.

Producto velocidad-potencia

Cuando, en una cierta aplicación, son importantes tanto el retardo de propagación como la disipación de potencia para la selección de la lógica que se debe utilizar, el producto velocidad-potencia es una base adecuada para la comparación de circuitos lógicos. Cuanto menor sea el producto velocidad-potencia, mejor. La unidad del producto velocidad-potencia es el picojulio (pJ). Por ejemplo, un HCMOS tiene un producto velocidad-potencia de 1,2 pJ a 100 kHz, mientras que un TTL de bajo consumo tiene un valor de 22 pJ.

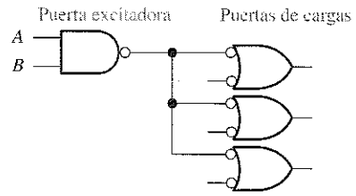
Carga y fan-out

Cuando la salida de una puerta lógica se conecta a una o más entradas de otras puertas, como se muestra en la Figura 15.10, se genera una carga en la puerta excitadora. Existe un límite para el número de entradas de carga que una cierta puerta puede excitar. Este límite se denomina **fan-out** de la puerta.

Carga CMOS. La carga en CMOS difiere de la de TTL en que en la lógica CMOS se utilizan transistores de efecto de campo (**FET**), que presentan una carga predomi-

nantemente capacitiva a la puerta excitadora, como ilustra la Figura 15.11. En este caso, las limitaciones vienen dadas por los tiempos de carga y descarga asociados con la resistencia de salida de la puerta excitadora y la capacitancia de entrada de las puertas de carga. Cuando la salida de la puerta excitadora está a nivel ALTO, la capacitancia de entrada de la puerta de carga se carga a través de la resistencia de salida de la puerta excitadora. Cuando la salida de la puerta excitadora está a nivel BAJO, la capacidad se descarga, como se indica en la Figura 15.11.

FIGURA 15.10
Carga de la salida de una puerta con las entradas de otras puertas.



Cuando se añaden más entradas de puertas de carga a la salida de la puerta excitadora, la capacitancia total aumenta, puesto que las capacitancias de entrada están en paralelo. Este aumento de la capacitancia aumenta los tiempos de carga y de descarga, por lo que se reduce la frecuencia máxima a la que la puerta puede funcionar. Por tanto, el fan-out de una puerta CMOS depende de la frecuencia de operación. Cuantas menos entradas de carga haya, mayor será la frecuencia máxima.

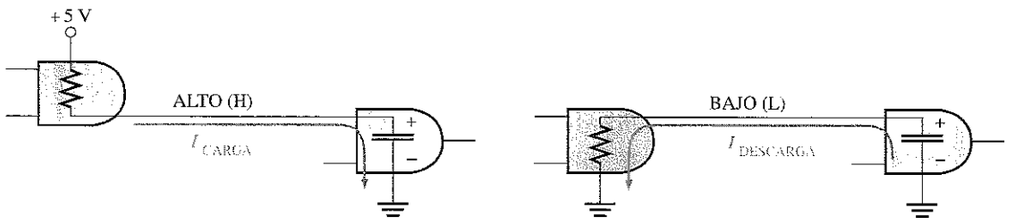


FIGURA 15.11
Carga capacitiva de una puerta CMOS.

Carga TTL. Una puerta excitadora TTL entrega (fuente) corriente a las entradas de las puertas de carga en el estado ALTO (I_{IH}) y absorbe corriente (sumidero) de las puertas de carga en el estado BAJO (I_{IL}). En la Figura 15.12 se ilustran de forma simplificada el funcionamiento como **fuente de corriente** y como **sumidero de corriente**, donde las resistencias representan las resistencias internas de entrada y salida de la puerta para ambas condiciones.

Cuantas más puertas de carga se conectan a la puerta excitadora, mayor es la carga de la misma. La corriente de fuente total aumenta con cada puerta de carga que se añade, como se muestra en la Figura 15.13. Al aumentar esta corriente, la caída de

tensión interna de la puerta excitadora aumenta, haciendo que la tensión de salida V_{OH} disminuya. Si se conecta un número excesivo de puertas de carga, la tensión V_{OH} cae por debajo de su valor mínimo, $V_{OH(mín)}$, reduciéndose el margen de ruido del nivel ALTO, lo que compromete el funcionamiento del circuito. También ocurre que, cuando la corriente de fuente aumenta, la disipación de potencia de la puerta excitadora aumenta.

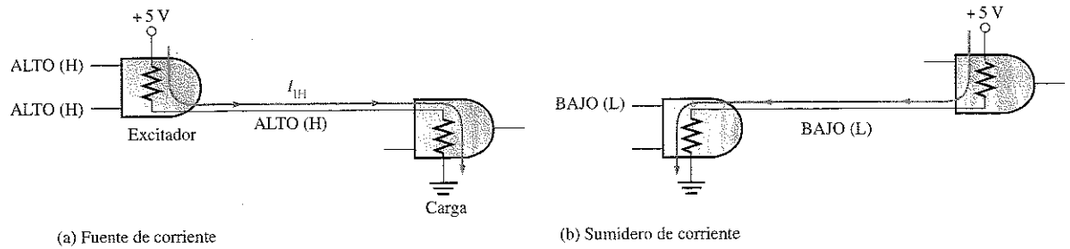


FIGURA 15.12
Fuente y sumidero de corriente en las puertas lógicas.

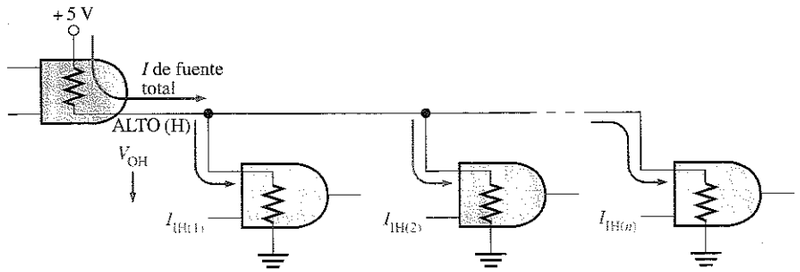


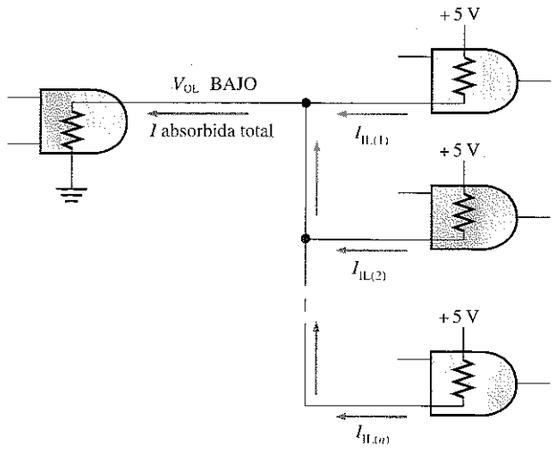
FIGURA 15.13
Carga TTL para el nivel ALTO.

El fan-out es el número máximo de entradas de carga que se pueden conectar sin afectar adversamente a las características de operación especificadas de la puerta. Por ejemplo, las TTL Schottky de bajo consumo (LS) tienen un fan-out de 20 unidades de carga. Una entrada de la misma familia lógica que la puerta excitadora se llama **unidad de carga.**

La corriente total de sumidero también aumenta con cada entrada de carga que se añade, como muestra la Figura 15.14. Al aumentar esta corriente, la caída de tensión interna de la puerta excitadora aumenta, haciendo que V_{OL} aumente. Si se añade un número demasiado grande de cargas, V_{OL} se hará mayor que $V_{OL(máx)}$, reduciéndose el margen de ruido para el nivel BAJO.

En TTL, la capacidad de la corriente de sumidero (estado de salida a nivel BAJO) es el factor más crítico en la determinación del fan-out.

FIGURA 15.14
Carga TTL en estado BAJO.



REVISIÓN DE LA SECCIÓN 15.1

1. Definir V_{IH}, V_{IL}, V_{OH} y V_{OL}.
2. ¿Qué es mejor, disponer de un valor de margen de ruido alto o bajo?
3. La puerta *A* tiene un retardo de propagación mayor que la puerta *B*. ¿Qué puerta puede trabajar a una frecuencia mayor?
4. ¿Cómo afecta una carga excesiva al margen de ruido de una puerta?

15.2 ■ CIRCUITOS CMOS

En esta sección se presenta la circuitería interna CMOS básica y su funcionamiento. Las siglas CMOS corresponden a complementary metal-oxide semiconductor (metal-óxido semiconductor complementario). El término complementario se refiere a la utilización de dos tipos de transistores en el circuito de salida. Se usan MOSFET (MOS field-effect transistor, transistor de efecto de campo MOS) de canal-n y de canal-p.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Identificar un MOSFET por su símbolo. ■ Explicar la acción de conmutación de un MOSFET. ■ Describir el funcionamiento básico de un circuito inversor CMOS. ■ Describir el funcionamiento básico de las puertas NAND y NOR CMOS. ■ Explicar el funcionamiento de una puerta CMOS con salida en drenador abierto. ■ Explicar el funcionamiento de las puertas CMOS de tres estados. ■ Enumerar las precauciones requeridas cuando se trabaja con dispositivos CMOS.

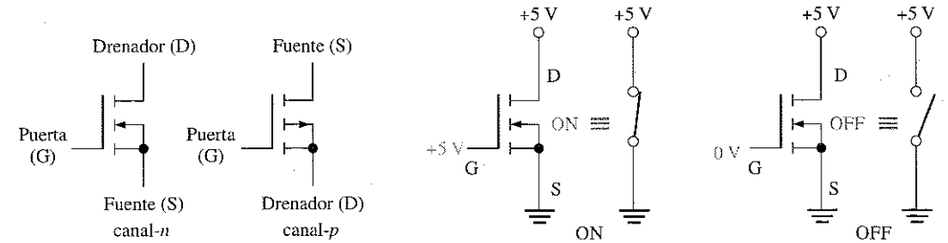
El MOSFET

Los transistores de efecto de campo de semiconductor de metal-óxido (**MOSFET**) son los elementos activos de conmutación de los circuitos CMOS. Estos dispositivos

difieren enormemente, tanto en la construcción como en el funcionamiento interno de los transistores bipolares utilizados en los circuitos TTL, pero, básicamente, su acción de conmutación es la misma. Idealmente, funcionan como interruptores abiertos o cerrados, dependiendo de la entrada.

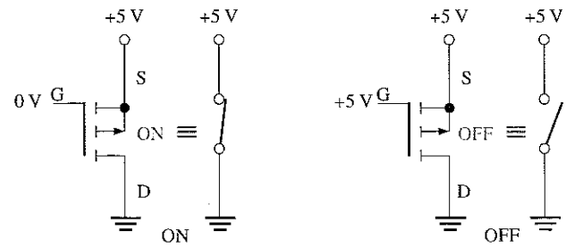
La Figura 15.15(a) presenta los símbolos para los MOSFET de canal-*n* y de canal-*p*. Como se puede ver, los tres terminales de un MOSFET son: **puerta, drenador** y **fuente**. Cuando la tensión de puerta de un MOSFET de canal-*n* es más positiva que la de fuente, el MOSFET conduce (*saturación*) e, idealmente, se comporta como un conmutador cerrado entre el drenador y la fuente. Cuando la tensión puerta-fuente es cero, el MOSFET no conduce (está al corte) e, idealmente, se comporta como un interruptor abierto entre el drenador y la fuente. Este modo de operación se ilustra en la Figura 15.15(b). Los MOSFET de canal-*p* funcionan con polaridades de tensión opuestas, como se muestra en la parte (c).

Algunas veces, se utiliza un símbolo simplificado para el MOSFET, como el mostrado en la Figura 15.16.



(a) Símbolos del MOSFET

(b) interruptor canal-*n*



(c) interruptor canal-*p*

FIGURA 15.15
Símbolos básicos y acción de conmutación de los MOSFET.

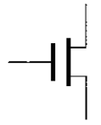


FIGURA 15.16
Símbolo simplificado del MOSFET.

Inversor CMOS

La lógica MOS complementaria (CMOS) utiliza pares complementarios MOSFET como elemento básico. Un par complementario emplea transistores MOSFET de canal-*p* y de canal-*n*, como muestra el circuito inversor de la Figura 15.17.

Cuando se aplica un nivel ALTO a la entrada, el MOSFET de canal-*p* Q_1 no conduce, y el MOSFET de canal-*n* Q_2 conduce (se satura), como se indica en la Figura 15.18(a). Esta condición hace que la salida se conecte a tierra a través de la resistencia de *conducción (on)* de Q_2, produciendo un nivel BAJO de salida. Cuando se aplica un nivel BAJO a la entrada, Q_1 se satura y Q_2 no, como se muestra en la Figura 15.18(b). Esta condición hace que la salida se conecte a $+V_{DD}$ (tensión de alimentación continua) a través de la resistencia de conducción de Q_1, produciendo un nivel ALTO de salida.

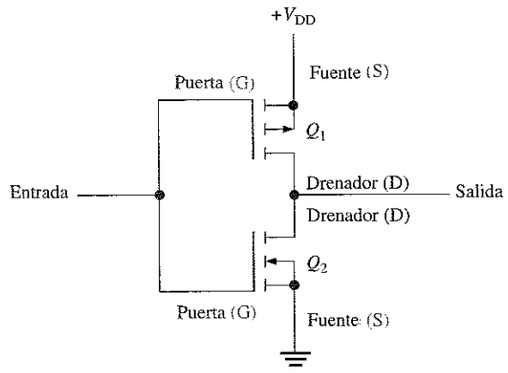


FIGURA 15.17
Circuito inversor CMOS.

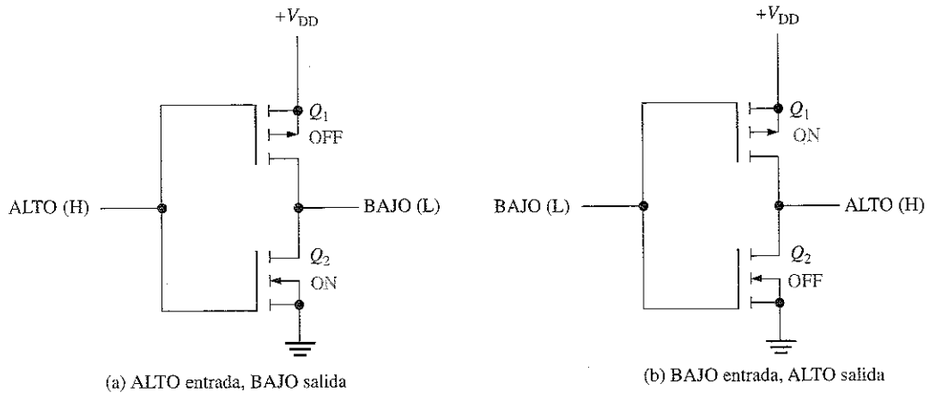
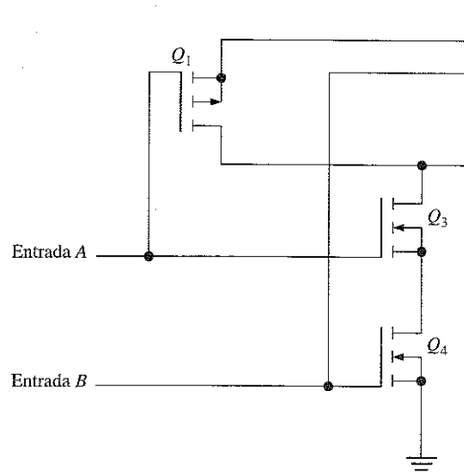


FIGURA 15.18
Funcionamiento de un inversor CMOS.

Puerta NAND CMOS

La Figura 15.19 muestra una puerta NAND CMOS con dos entradas. Obsérvese la disposición de los pares complementarios (dispositivos MOSFET de canal-*n* y canal-*p*).



A	B	Q_1	Q_2	Q_3	Q_4	X
L	L	S	S	C	C	H
L	H	S	C	C	S	H
H	L	C	S	S	C	H
H	H	C	C	S	S	L

C = corte (of)
S = saturación (on)
H = ALTO
L = BAJO

FIGURA 15.19
Circuito de una puerta CMOS NAND.

El funcionamiento de una puerta NAND CMOS es el siguiente:

- Cuando ambas entradas están a nivel BAJO, Q_1 y Q_2 se saturan, y Q_3 y Q_4 no. La salida se fuerza al nivel ALTO a través de las resistencias de conducción de Q_1 y Q_2, situadas en paralelo.
- Cuando la entrada *A* está a nivel BAJO y la entrada *B* esta a nivel ALTO, Q_1 y Q_4 se saturan, mientras que Q_2 y Q_3 no. La salida se fuerza a nivel ALTO a través de la baja resistencia de conducción de Q_1.
- Cuando la entrada *A* está a nivel ALTO y la entrada *B* está a nivel BAJO, Q_1 y Q_4 no conducen, mientras que Q_2 y Q_3 se saturan. La salida se fuerza a nivel ALTO a través de la baja resistencia de conducción de Q_2.
- Por último, cuando ambas entradas están a nivel ALTO, Q_1 y Q_2 no conducen, y Q_3 y Q_4 se saturan. En este caso, la salida se fuerza al nivel BAJO a través de las resistencias de conducción de Q_3 y Q_4 en serie con tierra.

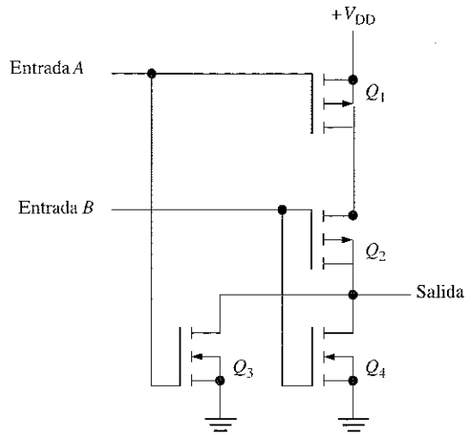
Puerta NOR CMOS

La Figura 15.20 muestra una puerta NOR CMOS con dos entradas. Obsérvese la disposición de los pares complementarios.

El funcionamiento de una puerta NOR CMOS es el siguiente:

- Cuando ambas entradas están a nivel BAJO, Q_1 y Q_2 se saturan, y Q_3 y Q_4 no. La salida se fuerza al nivel ALTO a través de las resistencias de conducción de Q_1 y Q_2, situadas en serie.

- Cuando la entrada A está a nivel BAJO y la entrada B esta a nivel ALTO, Q_1 y Q_4 se saturan, mientras que Q_2 y Q_3 no. La salida se fuerza a nivel BAJO a través de la resistencia de conducción de Q_4, conectada a tierra.
- Cuando la entrada A está a nivel ALTO y la entrada B esta a nivel BAJO, Q_1 y Q_4 no conducen y Q_2 y Q_3 se saturan. La salida se fuerza a nivel bajo a través de la resistencia de conducción de Q_3, conectada a tierra.
- Cuando ambas entradas están a nivel ALTO, Q_1 y Q_2 no conducen, mientras que Q_3 y Q_4 se saturan. La salida se fuerza al nivel BAJO a través de las resistencias de conducción de Q_3 y Q_4, en paralelo y conectadas a tierra.



A	B	Q_1	Q_2	Q_3	Q_4	X
L	L	S	S	C	C	H
L	H	S	C	C	S	L
H	L	C	S	S	C	L
H	H	C	C	S	S	L

C = corte (off)
S = saturación (on)
H = ALTO
L = BAJO

FIGURA 15.20
Circuito de una puerta CMOS NOR.

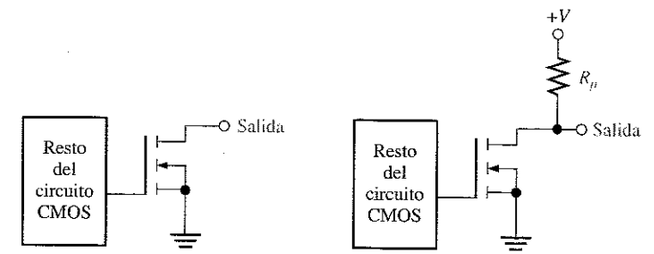
Puertas en drenador abierto

El término *drenador abierto* quiere decir que el drenador del transistor de salida no está conectado y debe conectarse externamente a V_{CC} a través de una carga. Las puertas en drenador abierto son la contrapartida CMOS de las puertas TTL en colector abierto. Un circuito de salida en drenador abierto es un único MOSFET de canal-n, como muestra la Figura 15.21(a). Debe utilizarse una **resistencia de pull-up**, como se indica en la parte (b), para producir un estado de salida ALTO. Las salidas en drenador abierto también pueden conectarse en configuración AND-cableada, concepto que se explica en la sección siguiente, dedicada a las puertas TTL.

Puertas CMOS triestado

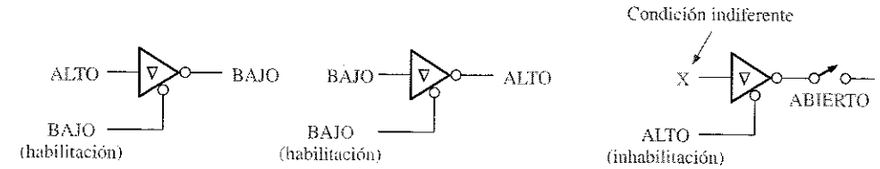
Las salidas triestado están disponibles tanto en la lógica CMOS como en la TTL. La salida **triestado** combina las ventajas de los circuitos totem-pole y en colector abierto. Como se recordará, los tres estados de salida son ALTO, BAJO y alta impedancia (**alta-Z**). Cuando se selecciona el modo normal de operación de nivel lógico, lo cual depende del estado de la entrada de habilitación, un circuito triestado funciona del

mismo modo que una puerta normal. Cuando se selecciona el modo de operación de alta impedancia, la salida se desconecta de forma efectiva del resto del circuito mediante la circuitería interna. La Figura 15.22 ilustra el funcionamiento de un circuito triestado. El triángulo invertido (∇) indica una salida triestado.



(a) Salida no conectada (b) Con resistencia de pull-up

FIGURA 15.21
Puertas CMOS en drenador abierto.



(a) Funcionamiento lógico normal (b) Estado de alta impedancia

FIGURA 15.22
Los tres estados de un circuito triestado.

La circuitería de un inversor CMOS triestado, como muestra la Figura 15.23, permite poner al corte a ambos transistores de salida, Q_1 y Q_2, a un mismo tiempo, desconectando la salida del resto del circuito.

Cuando la entrada de habilitación está a nivel BAJO, el dispositivo se activa para funcionamiento normal. Cuando la entrada de habilitación está a nivel ALTO, Q_1 y Q_2 se bloquean y el circuito está en estado de alta impedancia.

Precauciones para la manipulación de CMOS

Como ya hemos visto, todos los dispositivos CMOS son susceptibles de ser dañados por las descargas electrostáticas. Por tanto, debe trabajarse con ellos poniendo un especial cuidado. Téngase en cuenta las siguientes precauciones:

1. Todos los dispositivos CMOS se suministran embalados en espuma conductora para evitar la formación de cargas electrostáticas. Cuando se retiran de la espuma, los pines no deberían tocarse.

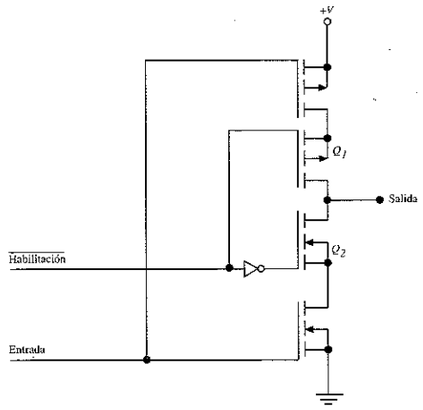
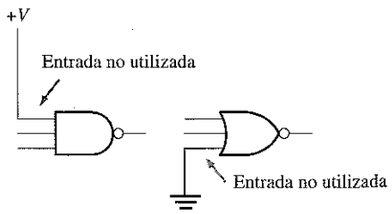


FIGURA 15.23
Inversor CMOS de tres estados.

2. Cuando se retiran del material protector, los dispositivos deben colocarse con los pines hacia abajo sobre una superficie a masa, tal como una placa metálica. No colocar los dispositivos CMOS sobre espumas de poliestireno o bandejas de plástico.
3. Todas las herramientas, equipos de pruebas y bancos de trabajo metálicos deberían estar conectados a tierra. En determinados entornos, la persona que trabaja con dispositivos CMOS debería conectar su muñeca a masa por medio de un cable y una resistencia serie de alto valor. Esta resistencia evita sufrir una descarga si la persona entra en contacto con una fuente de tensión.
4. No insertar los dispositivos CMOS (o cualquier otro dispositivo) en los zócalos de una tarjeta de circuito impreso con la alimentación conectada.
5. Todas las entradas no utilizadas deberían conectarse a la tensión de alimentación o a tierra, como se indica en la Figura 15.24. Si se deja en circuito abierto, una entrada puede adquirir carga electrostática y "flotar" en niveles impredecibles.

FIGURA 15.24
Tratamiento de las entradas CMOS no utilizadas.



6. Después del ensamblado de las tarjetas de circuito de impreso, éstas deben protegerse durante el almacenamiento o transporte insertando sus conectores en espuma conductora. Los pines de entrada y salida CMOS se pueden proteger también con resistencias de alto valor conectadas a masa.

REVISIÓN DE LA SECCIÓN 15.2

1. ¿Qué tipo de transistor se usa en la lógica CMOS?
2. ¿Qué significa el término *MOS complementario*?
3. ¿Por qué los dispositivos CMOS deben manipularse con cuidado?

15.3 ■ CIRCUITOS TTL

En esta sección se cubre el funcionamiento interno de los circuitos lógicos TTL con salidas totem-pole. También se tratará el funcionamiento de las puertas TTL con salida en colector abierto, y el de las puertas triestado.

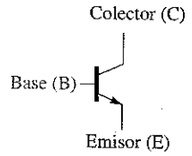
Al finalizar esta sección, el lector deberá ser capaz de:

■ Identificar mediante su símbolo al transistor de unión bipolar (BJT, bipolar junction transistor). ■ Describir la acción de conmutación de un BJT. ■ Describir el funcionamiento básico de un circuito inversor TTL. ■ Describir el funcionamiento básico de la puerta NAND TTL. ■ Explicar qué es una salida totem-pole. ■ Explicar el funcionamiento y utilización de una puerta TTL con salida en colector abierto. ■ Explicar el funcionamiento de una puerta con salida triestado.

El transistor de unión bipolar

El transistor de unión **bipolar (BJT)** es el elemento activo de conmutación utilizado en todos los circuitos TTL. La Figura 15.25 muestra el símbolo de un BJT *npn* con sus tres terminales: **base, emisor** y **colector**. Un BJT tiene dos **uniones**: la unión base-emisor y la unión base-colector.

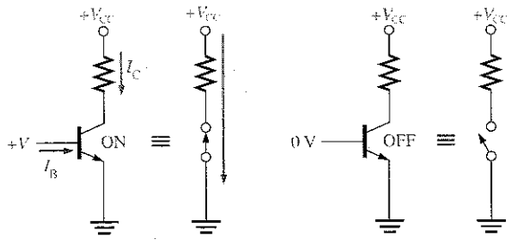
FIGURA 15.25
Símbolo del BJT.



La operación básica de conmutación es la siguiente: cuando la base está aproximadamente a 0,7 V por encima del emisor y se proporciona suficiente corriente a la base, el **transistor** conduce y entra en saturación. Idealmente, en saturación, el transistor actúa como un interruptor cerrado entre el colector y el emisor, como ilustra la Figura 15.26(a). Cuando la base está a menos de 0,7 V por encima del emisor, el transistor no conduce y actúa como un interruptor abierto entre el colector y el emisor, como muestra la parte (b). Resumiendo, un nivel ALTO en la base pone en conducción y satura (on) al transistor, por lo que actúa como un interruptor cerrado. Un nivel BAJO en la base bloquea (off) al transistor, el cual se comporta como un interruptor abierto. En la lógica TTL, existen algunos BJT con múltiples emisores.

Inversor TTL

La función lógica de un inversor o de cualquier tipo de puerta es siempre la misma, independientemente del tipo de tecnología de circuitos que se utilice. La Figura 15.27 muestra un circuito TTL estándar para un inversor. En esta figura, Q_1 es el transistor de acoplamiento de entrada y D_1 es el **diodo** de fijación del nivel de entrada (diodo "clamp"). El transistor Q_2 es el *divisor de fase* y la combinación de Q_3 y Q_4 forma el circuito de salida, a menudo denominado disposición **totem-pole**.



(a) Transistor saturado e interruptor equivalente ideal.

(b) Transistor bloqueado e interruptor equivalente ideal.

FIGURA 15.26
Conmutación ideal del BJT.

Cuando la entrada es un nivel ALTO, la unión base-emisor de Q_1 se **polariza en inversa**, y la unión base-colector se **polariza en directa**. Esta condición permite que la corriente atraviese R_1 y la unión base-colector de Q_1, llevando a Q_2 a la saturación. Como resultado, Q_2 excita a Q_3, y su tensión de colector, que es la de salida, es próxima al potencial de tierra. Por tanto, se obtiene una salida a nivel BAJO para una entrada a nivel ALTO. También, en este mismo instante, el colector de Q_2 está a un nivel de tensión suficientemente bajo como para mantener bloqueado a Q_4.

Cuando la entrada está a nivel BAJO, la unión base-emisor de Q_1 se polariza en directa y la unión base-colector se polariza en inversa, por lo que se genera una corriente a través de R_1 y de la unión base-emisor de Q_1. Un nivel BAJO proporciona un camino a tierra para la corriente. En la base de Q_2 no hay corriente, por lo que no conduce. El colector de Q_2 está a nivel ALTO, lo que pone en conducción a Q_4. El transistor Q_4 saturado proporciona un camino de baja resistencia desde V_{CC} hasta la salida; por tanto, un nivel BAJO en la entrada da lugar a un nivel ALTO en la salida. También, en este mismo instante, el emisor de Q_2 está a potencial de tierra, manteniendo bloqueado a Q_3.

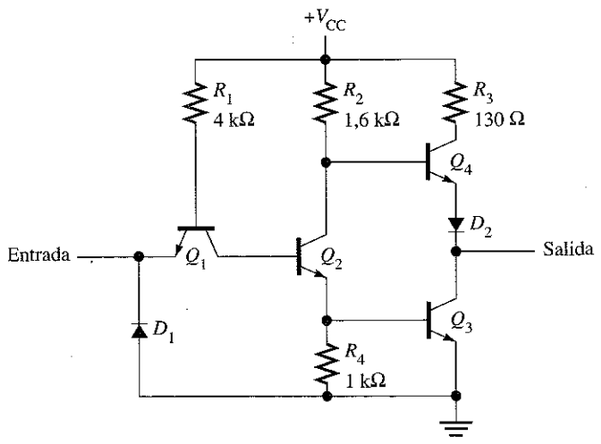


FIGURA 15.27
Circuito inversor TTL estándar.

El diodo D_1 del circuito TTL evita que los posibles picos negativos de tensión en la entrada puedan dañar a Q_1. El diodo D_2 asegura que Q_4 quede bloqueado cuando Q_2 conduce (entrada a nivel ALTO). En estas condiciones, la tensión de colector de Q_2 es igual a la tensión base-emisor V_{BE} de Q_3, más la tensión colector-emisor V_{CE} de Q_2. El diodo D_2 proporciona una caída adicional equivalente a V_{BE}, en serie con la unión base-emisor de Q_4, para asegurar su bloqueo cuando Q_2 conduce.

El funcionamiento del inversor TTL para los dos estados de entrada se ilustra en la Figura 15.28. En el circuito de la parte (a), la base de Q_1 está a 2,1 V respecto de masa, por lo que Q_2 y Q_3 se saturan. En el circuito de la parte (b), la base de Q_1 está aproximadamente a 0,7 V con respecto a tierra, lo que no es suficiente para poder saturar a Q_2 y Q_3.

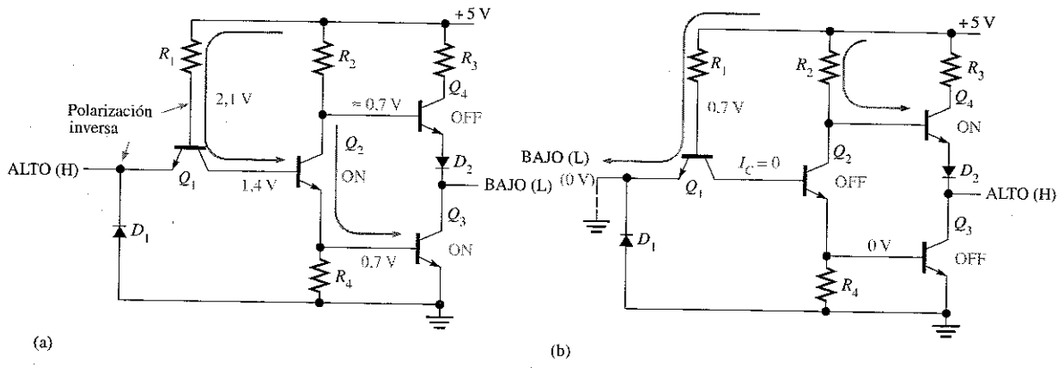


FIGURA 15.28
Funcionamiento de un inversor TTL.

Puerta NAND TTL

En la Figura 15.29 se muestra una puerta NAND TTL estándar de 2 entradas. Básicamente, es igual al circuito inversor, excepto en que Q_1 tiene un emisor de entrada adicional. En la tecnología TTL se utilizan transistores con múltiples emisores para los dispositivos de entrada. Estos transistores de múltiples emisores pueden compararse con la disposición de diodos mostrada en la Figura 15.30.

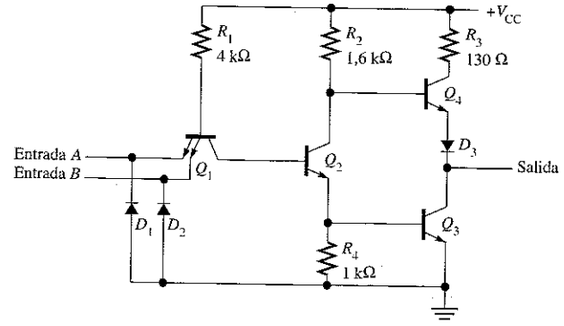


FIGURA 15.29
Circuito de una puerta NAND TTL.

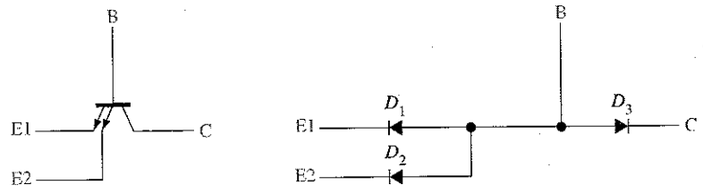


FIGURA 15.30
Equivalente con diodos de un transistor de múltiples emisores TTL.

Quizás sea más fácil entender el funcionamiento de este circuito reemplazando el transistor Q_1 de la Figura 15.29 por la combinación de diodos de la Figura 15.30. Un nivel BAJO en la entrada *A* o en la entrada *B* polariza en directa al respectivo diodo y en inversa a D_3 (unión base-colector de Q_1). Esto hace que Q_2 no conduzca, y da lugar a un nivel alto de salida, de la misma forma que se ha descrito para el inversor TTL. Por supuesto, un nivel bajo en ambas entradas producirá este mismo resultado.

Un nivel ALTO en ambas entradas polariza en inversa a ambos diodos y en directa al diodo D_3 (unión base-colector de Q_1). Esto hace que Q_2 entre en conducción y da lugar a un nivel BAJO de salida, del mismo modo que se ha descrito para el inversor TTL.

Debería reconocerse este modo de operación como el de la función NAND: la salida es un nivel BAJO si y sólo si todas las entradas están a nivel ALTO.

Puertas en colector abierto

Todas las puertas TTL descritas en los puntos anteriores tienen el circuito de salida totem-pole. Los circuitos integrados TTL disponen de otro tipo de salida, que es la salida en **colector abierto.** Este tipo de salida es comparable a la salida en drenador abierto de los CMOS. En la Figura 15.31(a) se presenta un inversor TTL estándar con salida en colector abierto. Otros tipos de puertas también están disponibles con este tipo de salidas.

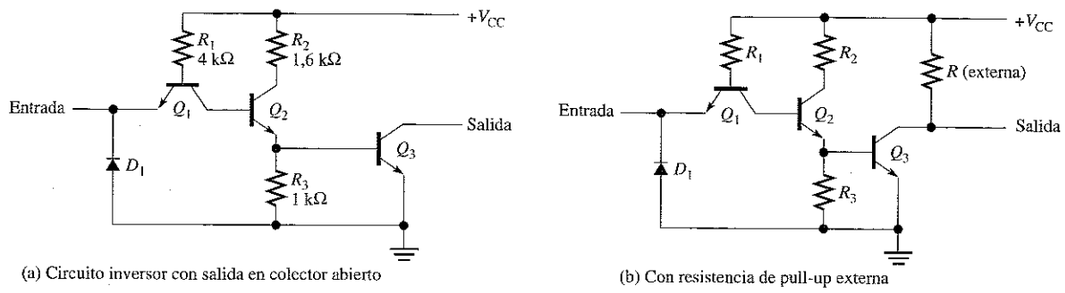
Obsérvese que la salida es el colector del transistor Q_3, sin nada conectado; de ahí el nombre de *colector abierto*. Para obtener los niveles lógicos ALTO y BAJO a la salida del circuito se conecta una resistencia de pull-up a V_{CC} desde el colector de Q_3, como muestra la Figura 15.31(b). Cuando Q_3 no conduce, la salida es llevada a V_{CC} a través de la resistencia externa. Cuando Q_3 se satura, la salida se lleva a un potencial próximo a tierra a través del transistor saturado.

El símbolo estándar ANSI/IEEE que designa la salida en colector abierto para un inversor se muestra en la Figura 15.32, y es el mismo que para una salida en drenador abierto.

Puertas TTL con salida triestado

La Figura 15.33 muestra el circuito básico de un inversor triestado TTL. Cuando la entrada de habilitación está a nivel BAJO, Q_2 no conduce y el circuito de salida funciona en la configuración totem-pole normal, en la que el estado de salida depende del

estado de entrada. Cuando la entrada de habilitación está a nivel ALTO, Q_2 conduce. Debido a ello, en el segundo emisor de Q_1 se produce un nivel BAJO, haciendo que Q_3 y Q_5 se bloqueen y el diodo D_1 se polarice en directa, lo que hace que Q_4 se bloquee también. Cuando ambos transistores totem-pole se bloquean, actúan como un circuito abierto y la salida está por completo desconectada de la circuitería interna, como muestra la Figura 15.34.



(a) Circuito inversor con salida en colector abierto

(b) Con resistencia de pull-up externa

FIGURA 15.31
Inversor TTL estándar con salida en colector abierto.

FIGURA 15.32
Símbolo de colector abierto en un inversor.

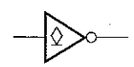
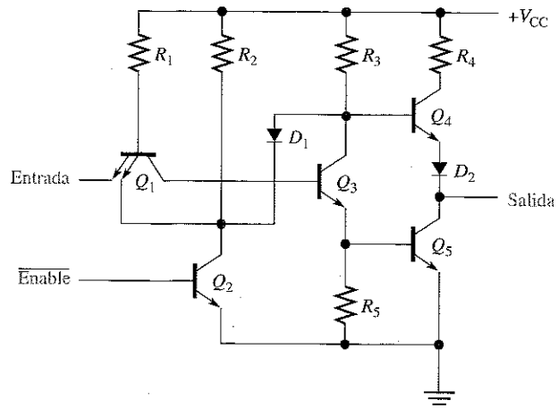


FIGURA 15.33
Circuito inversor de tres estados básico.



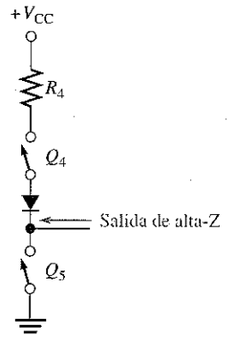
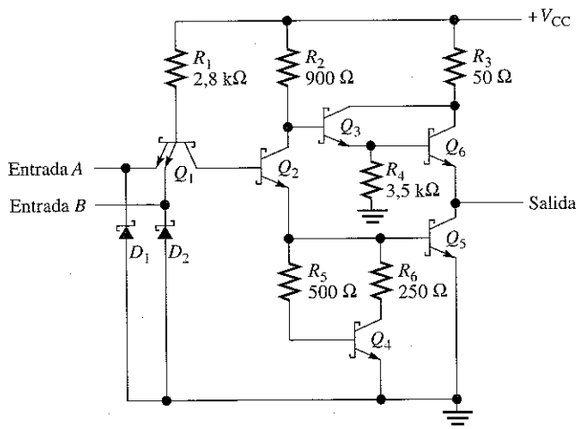


FIGURA 15.34
Circuito equivalente de la salida triestado en el estado de alta-Z.

TTL Schottky

Anteriormente se ha estudiado el circuito básico de la puerta NAND TTL. Es un sumidero de corriente que absorbe corriente de la carga cuando el estado de salida es un nivel BAJO, y suministra una corriente despreciable a la carga cuando el estado de salida es ALTO. La mayor parte de la lógica TTL utilizada actualmente es de tipo TTL Schottky, que proporciona un muy rápido tiempo de conmutación mediante la incorporación de diodos Schottky, que evitan que los transistores entren en saturación, disminuyendo el tiempo que tarda el transistor en entrar y salir de conducción. La Figura 15.35 muestra un circuito de puerta NAND Schottky. Obsérvense los símbolos de los transistores Schottky y de los diodos Schottky. Los dispositivos Schottky se designan mediante una *S* en su identificador de dispositivo. Otros tipos de TTL Schottky son la lógica TTL Schottky de bajo consumo, que se designa mediante las letras LS; la lógica Schottky avanzada, que se designa mediante AS; Schottky de bajo consumo avanzada, que se especifica mediante ALS; y la serie rápida, que se designa mediante la letra F.

FIGURA 15.35
Puerta NAND TTL Schottky.



REVISIÓN DE LA SECCIÓN 15.3

1. Verdadero o falso: un BJT *npn* se satura cuando la base es más negativa que el emisor.
2. En términos de conmutación, ¿qué representan los estados de saturación y bloqueo de un BJT?
3. ¿Cuáles son los dos principales tipos de salida en los circuitos TTL?
4. Explicar en qué difiere la lógica de tres estados de la lógica normal de dos estados.

15.4 ■ CONSIDERACIONES PRÁCTICAS SOBRE EL USO DE TTL

Aunque CMOS es la tecnología de circuitos integrados predominante en la industria y en las aplicaciones comerciales, la tecnología TTL todavía se emplea. En las aplicaciones para la enseñanza, normalmente se prefiere la tecnología TTL porque no tiene las restricciones de manipulación que tiene la tecnología CMOS, debidas a las descargas electrostáticas. Por esto, vamos a examinar varias consideraciones prácticas para el uso y aplicación de los circuitos TTL, utilizando la lógica TTL estándar como ilustración.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Describir los conceptos de fuente y sumidero de corriente. ■ Utilizar un circuito en colector abierto para la operación AND-cableada. ■ Describir los efectos de conectar dos o más salidas totem-pole. ■ Utilizar las puertas en colector abierto para excitar diodos LED y lámparas. ■ Explicar qué se hace con las entradas TTL no utilizadas.

Fuente y sumidero de corriente

En la Sección 15.1 se han introducido los conceptos de fuente y sumidero de corriente. Ahora que ya estamos familiarizados con las configuraciones de los circuitos con salida totem-pole utilizados en la lógica TTL, vamos a ver más en detalle la operación como fuente y sumidero de corriente.

La Figura 15.36 muestra un inversor TTL estándar, con una salida totem-pole conectada a la entrada de otro inversor TTL. Cuando la puerta excitadora tiene un estado de salida ALTO, actúa como fuente de corriente para la carga, como muestra la Figura 15.36(a). La entrada de la puerta de carga es como un diodo polarizado en inversa, por lo que la carga prácticamente no requiere corriente. Realmente, puesto que es una entrada no ideal, habrá un máximo de 40 μA de corriente desde la salida totem-pole del excitador a la entrada de la puerta de carga.

Cuando la puerta excitadora está en el estado de salida BAJO, el excitador actúa como sumidero de corriente de la carga, como muestra la Figura 15.36(b). Esta corriente es 1,6 mA como máximo para los dispositivos TTL estándar y en las **hojas de características** se indica con un valor negativo, puesto que *sale* de la entrada.

EJEMPLO 15.3 Cuando una puerta NAND TTL estándar excita cinco entradas TTL, ¿cuánta corriente entrega (fuente) la salida del excitador y cuánta absorbe (sumidero)?

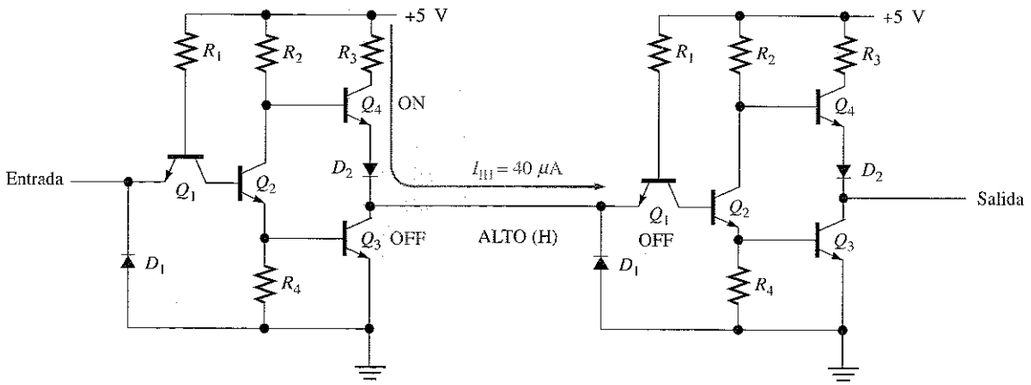
Solución. La corriente total de fuente es (para el estado de salida ALTO):

$$I_{IH(máx)} = 40\ \mu A \text{ por entrada}$$
$$I_{T(fuente)} = (5 \text{ entradas})(40\ \mu A/\text{entrada}) = 5(40\ \mu A) = \mathbf{200\ \mu A}$$

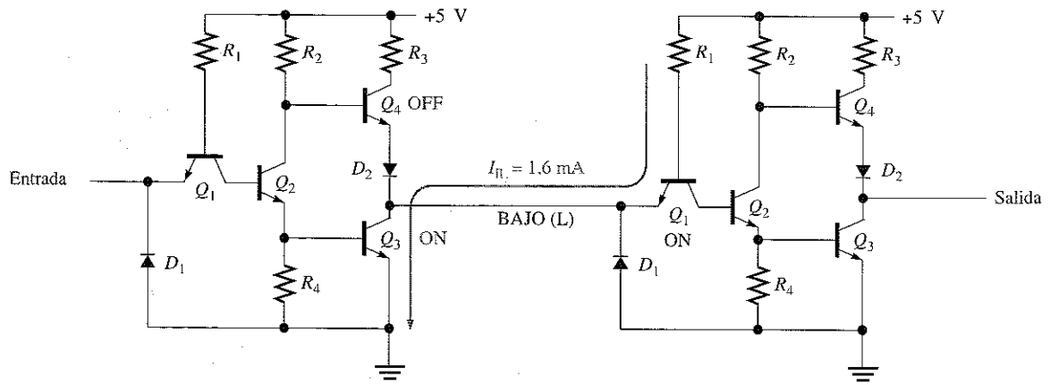
La corriente total absorbida (en el estado de salida BAJO) es:

$$I_{IL(máx)} = -1{,}6 \text{ mA por entrada}$$
$$I_{T(absorbida)} = (5 \text{ entradas})(-1{,}6 \text{ mA/entrada}) = 5(-1{,}6 \text{ mA}) = \mathbf{-8{,}0\ mA}$$

Problema relacionado. Repetir los cálculos para una puerta TTL LS. Utilizar una hoja de características.



(a) Fuente de corriente (el valor de I_{IH} es máximo)



(b) Sumidero de corriente (el valor de I_{IL} es máximo)

FIGURA 15.36
Operación de un circuito TTL como fuente y sumidero de corriente.

EJEMPLO 15.4 Utilizando las hojas de características del Apéndice A, determinar el fan-out de la puerta NAND 7400.

Solución. De acuerdo con la hoja de características, los parámetros de corriente son:

$$I_{IH(máx)} = 40\ \mu A \qquad I_{IL(máx)} = 1{,}6\ mA$$
$$I_{OH(máx)} = -400\ \mu A \qquad I_{OL(máx)} = 16\ mA$$

El fan-out para el estado de salida ALTO se obtiene del siguiente modo: la corriente $I_{OH(máx)}$ es la corriente máxima que la puerta puede entregar a la carga. Cada entrada de carga requiere una $I_{IH(máx)}$ de 40 μA. El fan-out del estado ALTO es

$$\left|\frac{I_{OH(máx)}}{I_{IH(máx)}}\right| = \frac{400\mu A}{40\mu A} = \mathbf{10}$$

Para el estado de salida a nivel BAJO, el fan-out se calcula como sigue: $I_{OL(máx)}$ es la máxima corriente que la puerta puede absorber. Cada entrada de carga requiere una $I_{IL(máx)}$ de 1,6 mA. El fan-out del estado BAJO es

$$\left|\frac{I_{OL(máx)}}{I_{IL(máx)}}\right| = \frac{16mA}{1{,}6mA} = \mathbf{10}$$

En este caso, el fan-out del estado alto y del estado bajo son iguales.

Problema relacionado. Determinar el fan-out de una puerta NAND 74LS00.

Utilización de puertas en colector abierto para la operación AND-cableada

Las salidas de las puertas en colector abierto se pueden conectar juntas para formar lo que se denomina una configuración *AND-cableada*. La Figura 15.37 ilustra cómo se conectan cuatro inversores para generar una puerta negativa-AND de 4 entradas. En todos los circuitos de AND-cableada, se requiere una resistencia de pull-up externa, R_p.

Cuando una (o más) de las entradas de los inversores están a nivel ALTO, la salida X pasa a nivel BAJO, puesto que un transistor de salida conduce y actúa como un interruptor cerrado conectado a tierra, como ilustra la Figura 15.38(a). En este caso, sólo la entrada de un inversor está a nivel ALTO, pero es suficiente para forzar la salida al nivel BAJO por medio de la saturación del transistor Q_1, tal y como se indica.

Para que la salida X sea un nivel ALTO, *todas* las entradas de los inversores deben estar a nivel BAJO, de modo que los transistores de salida en colector abierto estén bloqueados, como se indica en la Figura 15.38(b). Cuando se produce esta condición, se fuerza a la salida X al nivel ALTO por medio de la resistencia de pull-up. Luego la salida X es un nivel ALTO sólo cuando *todas* las entradas están a nivel BAJO. Por tanto, se trata de una función negativa-AND, como expresa la siguiente ecuación:

$$X = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$$

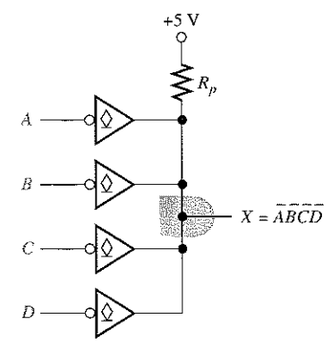
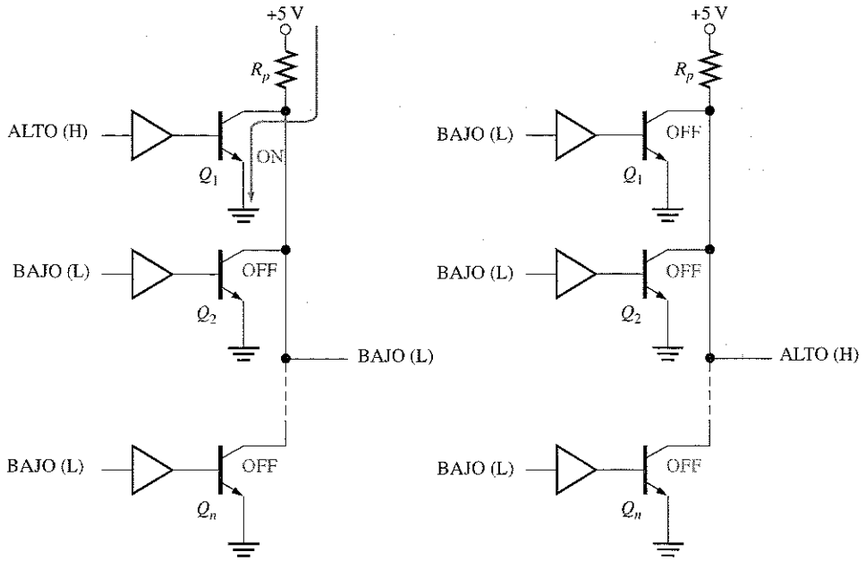


FIGURA 15.37
Configuración AND-cableada con cuatro inversores.



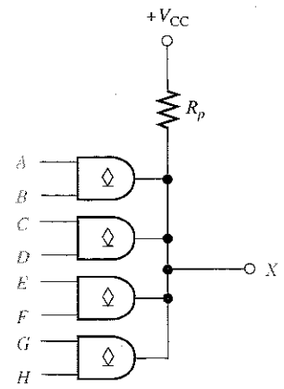
(a) Cuando uno o más transistores de salida se saturan, la salida es un nivel BAJO

(b) Cuando todos los transistores de salida están bloqueados, la salida es un nivel ALTO

FIGURA 15.38
Operación negativa-AND con inversores con salida en colector abierto cableada.

EJEMPLO 15.5 Escribir la expresión de salida de la configuración AND-cableada con puertas AND en colector abierto de la Figura 15.39.

FIGURA 15.39



Solución. La expresión de salida es

$$X = ABCDEFGH$$

La conexión AND-cableada de las cuatro puertas AND de dos entradas da lugar a una puerta AND de 8 entradas.

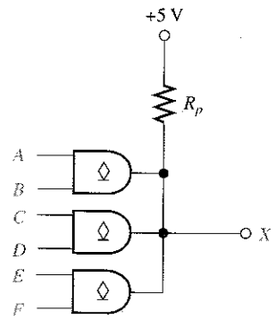
Problema relacionado. Determinar la expresión de salida si, en la Figura 15.39, se usan puertas NAND.

EJEMPLO 15.6 Tres puertas AND en colector abierto se conectan en una configuración AND-cableada como la mostrada en la Figura 15.40. Suponer que el circuito AND-cableada se excita mediante cuatro entradas TTL estándar (–1,6 mA cada una).

(a) Escribir la expresión lógica de X.

(b) Determinar el valor mínimo de R_p, si $I_{OL(máx)}$ para cada puerta es 30 mA y $V_{OL(máx)}$ es 0,4 V.

FIGURA 15.40



Solución

(a) $X = ABCDEF$

(b) 4 (1,6 mA) = 6,4 mA

$$I_{R_p} = I_{OL(máx)} - 6{,}4\text{ mA} = 30\text{ mA} - 6{,}4\text{ mA} = 23{,}6\text{ mA}$$

$$R_p = \frac{V_{CC} - V_{OL(máx)}}{I_{R_p}} = \frac{5\text{ V} - 0{,}4\text{ V}}{23{,}6\text{ mA}} = \mathbf{195\ \Omega}$$

Problema relacionado. Diseñar el circuito AND-cableada para obtener una función AND de 10 entradas utilizando circuitos 74LS09, cuádruple puerta AND de 2 entradas.

Conexión de las salidas totem-pole

Las salidas totem-pole no se pueden conectar juntas, porque dicha conexión produce una corriente excesiva y daría lugar a daños en los dispositivos. Por ejemplo, en la Figura 15.41, cuando los transistores Q_1 del dispositivo A y Q_2 del dispositivo B conducen, la salida del dispositivo queda cortocircuitada a tierra a través del transistor Q_2 del dispositivo B.

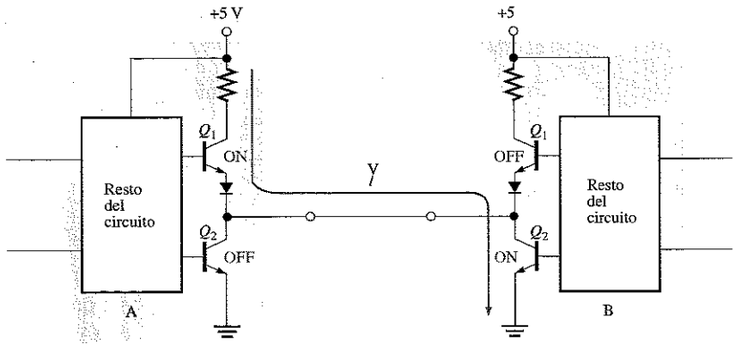
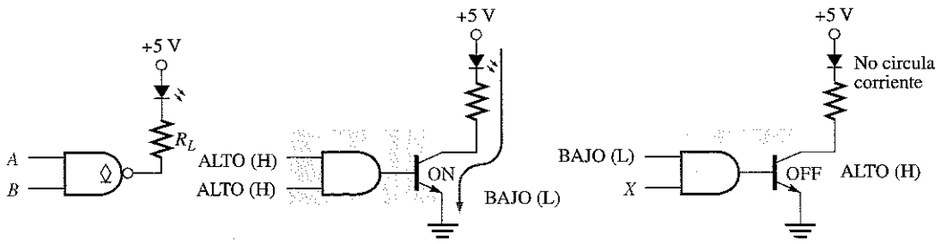


FIGURA 15.41
Salidas totem-pole conectadas juntas. Esta conexión puede generar una corriente excesiva a través del transistor Q_1 del dispositivo A y del transistor Q_2 del dispositivo B, y no debería emplearse nunca.

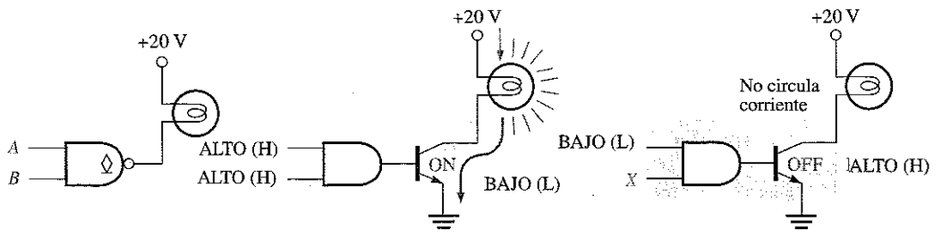
Buffer/excitador en colector abierto

Un circuito TTL con una salida totem-pole tiene limitada la cantidad de corriente que puede absorber en el estado BAJO ($I_{OL(máx)}$), a 16 mA para la lógica TTL estándar y a 8 mA para la lógica TTL LS. En muchas aplicaciones especiales, una puerta tiene que excitar a dispositivos externos tales como diodos LED, lámparas o relés, que pueden requerir más corriente que la que pueden proporcionar estos dispositivos.

Generalmente, para excitar diodos LED, lámparas o relés, se utilizan circuitos con salidas en colector abierto, debido a su mayor capacidad de manejo de corriente y tensión. Sin embargo, las salidas totem-pole pueden utilizarse mientras que la corriente de salida requerida por el dispositivo externo no exceda el valor que el excitador TTL pueda absorber.



(a) Excitación de un LED



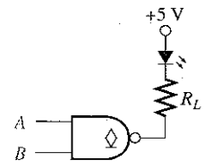
(b) Excitación de una lámpara

FIGURA 15.42
Aplicaciones de los excitadores en colector abierto.

En una puerta TTL en colector abierto, el colector del transistor de salida se conecta al **LED** o a la lámpara incandescente, como muestra la Figura 15.42. En la parte (a), se utiliza una resistencia de limitación para mantener la corriente por debajo de la corriente máxima del LED. Cuando la salida de la puerta es un nivel BAJO, el transistor de salida opera como sumidero de corriente, y el LED se enciende. El LED se apaga cuando el transistor de salida no conduce y la salida está a nivel alto. Una puerta buffer en colector abierto típica puede absorber hasta 40 mA. En la parte (b) de la figura, la lámpara no requiere una resistencia de limitación, ya que el filamento es resistivo. Típicamente, en colector abierto pueden utilizarse hasta +30 V, dependiendo de la familia lógica en particular.

EJEMPLO 15.7 Determinar la resistencia de limitación R_L en el circuito en colector abierto de la Figura 15.43, si la corriente del LED debe ser 20 mA. Suponer una caída de tensión de 1,5 V en el LED cuando está polarizado en directa y una tensión de salida para el estado BAJO de 0,1 V en la salida de la puerta.

FIGURA 15.43



Solución

$$V_{R_L} = 5\text{ V} - 1{,}5\text{ V} - 0{,}1\text{ V} = 3{,}4\text{ V}$$

$$R_L = \frac{V_{R_L}}{I} = \frac{3{,}4\text{ V}}{20\text{ mA}} = \mathbf{170\Omega}$$

Problema relacionado. Determinar el valor de la resistencia de limitación R_L, si el LED requiere 35 mA.

Entradas TTL no utilizadas

Una entrada desconectada de una puerta TTL actúa como un nivel lógico ALTO, ya que una entrada en abierto da lugar a una unión de emisor polarizada en inversa en el transistor de entrada, al igual que un nivel ALTO. En la Figura 15.44 se ilustra este efecto. Sin embargo, debido a la sensibilidad al ruido, es mejor no dejar las entradas no utilizadas desconectadas (en abierto). Existen varias alternativas para tratar este tipo de entradas no utilizadas.

Entradas unidas. El método más común de manipular las entradas no utilizadas de una puerta es conectarlas a una entrada que sí se use de la misma puerta. Para las puertas NAND y AND, todas las entradas unidas cuentan como una unidad de carga para

el estado BAJO; pero en las puertas OR y NOR, cada entrada conectada a otra entrada cuenta como una carga adicional en el estado BAJO. En el estado ALTO, cada entrada unida cuenta como una unidad de carga adicional para todos los tipos de puertas TTL. En la Figura 15.45(a) se presentan dos ejemplos de conexión de dos entradas no utilizadas a una entrada que sí se utiliza.

Las puertas AND y NAND representan una única unidad de carga, independientemente de cuántas entradas se conecten juntas, mientras que para las puertas OR y NOR, cada entrada unida representa una unidad de carga. Esto se debe a que la puerta NAND utiliza un transistor de entrada de múltiples emisores, por lo que, independientemente de cuántas entradas estén a nivel BAJO, la corriente total para el estado BAJO está limitada a un valor fijo. La puerta NOR utiliza un transistor distinto para cada entrada; luego para el estado BAJO, la corriente total es la suma de las corrientes de todas las entradas unidas.

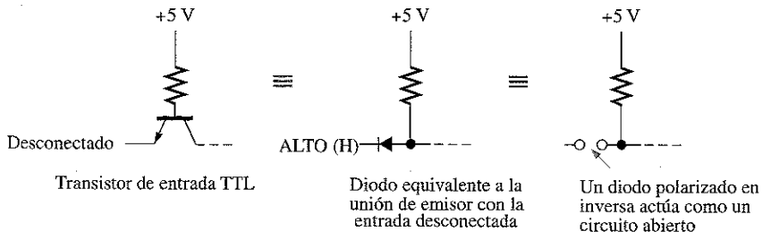


FIGURA 15.44
Comparación de una entrada TTL en abierto y una entrada a nivel ALTO.

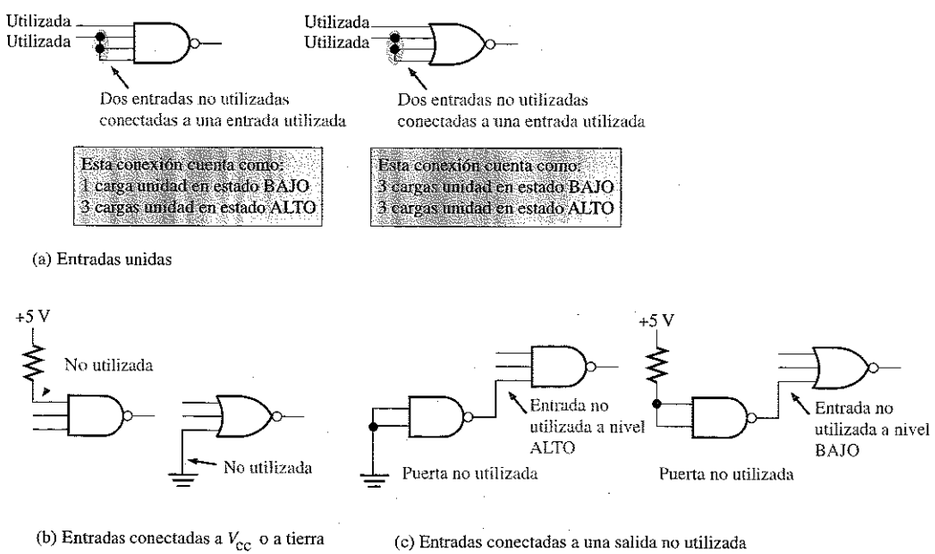


FIGURA 15.45
Métodos para tratar las entradas TTL no utilizadas.

Entradas conectadas a V_{CC} o a tierra. Las entradas no utilizadas de las puertas AND y NAND se pueden conectar a V_{CC} a través de una resistencia de 1,0 kΩ. Esta conexión fuerza a las entradas no utilizadas a un nivel ALTO. Las entradas no utilizadas de las puertas OR y NOR se conectan a tierra. Estos métodos se ilustran en la Figura 15.45(b).

Entradas conectadas a una salida no utilizada. Un tercer método de terminación de las entradas no utilizadas puede ser adecuado en ciertos casos, cuando se dispone de una puerta o un inversor no utilizado. La salida de la puerta no utilizada debe estar a un nivel ALTO constante para las entradas AND y NAND no utilizadas, y a un nivel BAJO constante para las entradas OR y NOR, como muestra la Figura 15.45(c).

REVISIÓN DE LA SECCIÓN 15.4

1. ¿En qué estado de salida absorbe corriente de la carga un circuito TTL?
2. ¿Por qué un circuito TTL entrega menos corriente a una carga TTL que la que absorbe?
3. ¿Por qué los circuitos TTL con salidas totem-pole no se pueden conectar entre sí?
4. ¿Qué tipo de circuitos TTL deben utilizarse para obtener una configuración AND-cableada?
5. ¿Qué tipo de circuitos TTL debería utilizar para excitar una lámpara.
6. Una entrada TTL desconectada actúa como un nivel BAJO (verdadero o falso).

15.5 ■ COMPARACIÓN DE LAS PRESTACIONES CMOS Y TTL

En esta sección, se van a comparar las principales prestaciones y características de funcionamiento de ciertas familias CMOS con las de las principales familias TTL y con BiCMOS.

Al finalizar esta sección, el lector deberá ser capaz de:

- Comparar los dispositivos TTL (bipolar), BiMOS y CMOS en términos de retardo de propagación, frecuencia de reloj máxima, disipación de potencia y capacidad de excitación.

En el pasado, las características superiores de los dispositivos TTL (bipolar) frente a los CMOS eran su relativamente alta velocidad y la capacidad de corriente de salida. Ahora, estas ventajas de la tecnología TTL han disminuido hasta el punto en que los dispositivos CMOS, a menudo, son iguales o superiores en muchas áreas y han llegado a ser la tecnología dominante de circuitos integrados, aunque la lógica TTL todavía está ampliamente disponible y en uso, como ya sabemos. Existe una familia de circuitos integrados, BiCMOS, que combina la lógica CMOS con la circuitería de salida TTL, en un esfuerzo por combinar las ventajas de ambas tecnologías.

La Tabla 15.1 proporciona una comparación de los parámetros de funcionamiento de varias familias lógicas de circuitos integrados.

TABLA 15.1
Comparación de los principales parámetros de funcionamiento de varias familias de circuitos integrados 74XX.

	Bipolar (TTL)			BiCMOS	CMOS					
					5 V			3,3 V		
	F	LS	ALS	ABT	HC	AC	AHC	LV	LVC	ALVC
Velocidad Retardo de propagación de puerta, t_p (ns)	3,3	10	7	3,2	7	5	3,7	9	4,3	3
Frecuencia máxima de reloj (MHz)	145	33	45	150	50	160	170	90	100	150
Disipación de potencia/puerta Bipolar: 50% dc (mW) CMOS: reposo (μW)	6	2,2	1,4	17	2,75	0,55	2,75	1,6	0,8	0,8
Excitación de salida I_{OL} (mA)	20	8	8	64	4	24	8	12	24	24

REVISIÓN DE LA SECCIÓN 15.5

1. ¿Qué es un circuito BiCMOS?
2. En general, ¿cuál es la principal ventaja de la tecnología CMOS sobre la bipolar (TTL)?

15.6 ■ CIRCUITOS ECL

ECL (Emitter-Coupled Logic, lógica de emisor acoplado), al igual que la TTL, es una tecnológica bipolar. El circuito ECL típico está formado por un circuito de entrada amplificador diferencial, un circuito de polarización y salidas de tipo seguidor de emisor. ECL es mucho más rápida que TTL, ya que los transistores no funcionan en saturación y se emplea en aplicaciones de alta velocidad especializadas.

Al finalizar esta sección, el lector deberá ser capaz de:

■ Explicar en qué difiere la tecnología ECL de TTL y CMOS. ■ Explicar las ventajas y desventajas de la lógica ECL.

En la Figura 15.46(a) se presenta una puerta OR/NOR **ECL**. Las salidas en seguidor de emisor proporcionan la función lógica OR y su complementaria NOR, como se indica en la Figura 15.46(b).

Debido a la baja impedancia de salida del seguidor de emisor y a la alta impedancia de entrada del amplificador diferencial, es posible trabajar con un alto fan-out. En este tipo de circuito, no es posible la saturación. Esta falta de saturación da lugar a un muy alto consumo de potencia y a recorridos de tensión limitados (menores de 1 V), pero permite la conmutación a alta frecuencia.

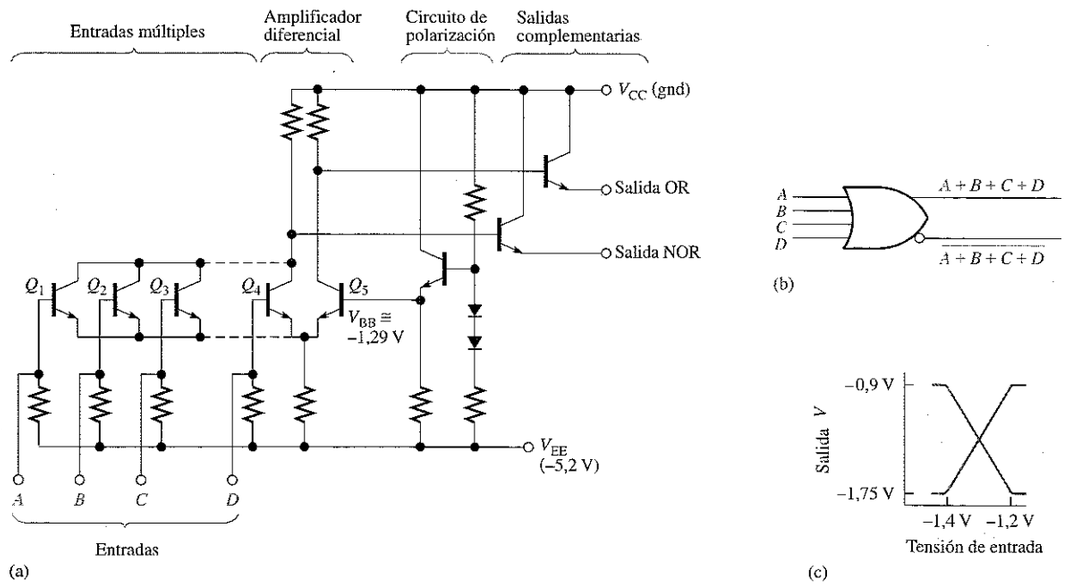


FIGURA 15.46
Circuito de una puerta OR/NOR ECL.

Normalmente, el pin V_{CC} se conecta a tierra y el pin V_{EE} se conecta a –5,2 V con respecto a la tensión de alimentación, para obtener un funcionamiento óptimo. Obsérvese en la Figura 15.46(c) que la salida varía desde –1,75 V para el nivel BAJO a -0,9 V para el nivel ALTO, con respecto a masa. En la lógica positiva, un 1 es un nivel ALTO (menos negativo) y un 0 es un nivel BAJO (más negativo).

Margen de ruido

Como ya sabemos, el margen de ruido de una puerta es la medida de su inmunidad a las fluctuaciones de tensión indeseadas (ruido). Típicamente, los circuitos ECL tienen márgenes de ruido comprendidos, aproximadamente, entre 0,2 V y 0,25 V. Estos márgenes son menores que en la lógica TTL, y hacen a la lógica ECL poco fiable en entornos de alto ruido.

Comparación de ECL con TTL y CMOS

La Tabla 15.2 muestra una comparación de los parámetros clave de funcionamiento para las familias F y AHC, por un lado, y dos familias ECL , 10H y E-lite, por otro.

TABLA 15.2
Comparación de los parámetros de funcionamiento de dos series ECL con las familias F y AHC.

	Bipolar (TTL)	CMOS	Bipolar (ECL)	
	F	AHC	10H	E-lite
Velocidad Retardo de propagación de puerta, t_p (ns)	3,3	3,7	1	0,22
Frecuencia máxima de reloj (MHz)	145	170	330	2800
Disipación de potencia/puerta Bipolar: 50% dc CMOS: reposo	8,9 mW	2,5 μW	25 mW	73 mW

REVISIÓN DE LA SECCIÓN 15.6

1. ¿Cuál es la principal ventaja de ECL sobre TTL?
2. Citar dos desventajas de la lógica ECL respecto a la TTL.

15.7 ■ PMOS, NMOS Y E²CMOS

Los circuitos PMOS y NMOS se utilizan en muchas funciones LSI, tales como registros de desplazamiento, grandes memorias y microprocesadores. Su uso se debe al bajo consumo de potencia y la pequeña área de chip que requieren los transistores MOS. E²CMOS se usa en dispositivos PLD reprogramables.

Al finalizar esta sección, el lector deberá ser capaz de:

- Describir una puerta PMOS básica.
- Describir una puerta NMOS básica.
- Describir una celda E²CMOS básica.

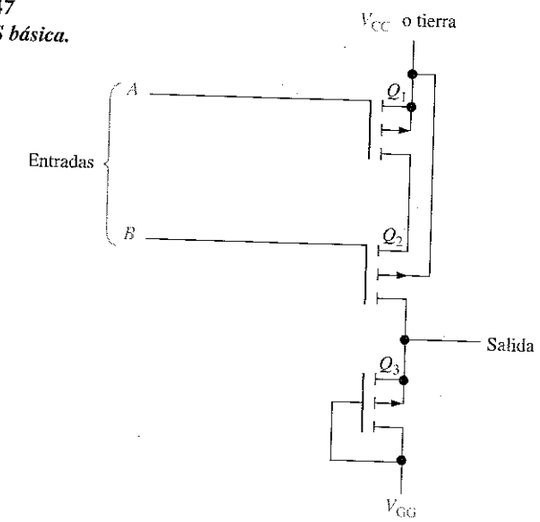
PMOS

Una de las primeras tecnologías para producir circuitos **MOS** de alta densidad fue la **PMOS**. Utilizaba transistores MOS de canal-*p* en modo de acumulación para producir los elementos de puerta básicos. La Figura 15.47 muestra una puerta PMOS que genera la función NOR en lógica positiva.

El funcionamiento de la puerta PMOS es el siguiente: la tensión de alimentación V_{GG} es una tensión negativa y V_{CC} es una tensión positiva o masa (0 V). El transistor Q_3 está polarizado permanentemente para crear una resistencia de drenador-fuente constante. Su único propósito es funcionar como resistencia limitadora de corriente. Si se aplica un nivel ALTO (V_{CC}) a la entrada A o B, entonces Q_1 o Q_2 no conducen, y la salida se fuerza a una tensión próxima a V_{GG}, lo que representa un nivel BAJO.

Cuando se aplica una tensión a nivel BAJO (V_{GG}) a ambas entradas A y B, Q_1 y Q_2 entran en conducción. Esto hace que la salida pase a nivel ALTO (próximo a V_{CC}). Puesto que se produce un nivel BAJO de salida cuando cualquier entrada o ambas están a nivel ALTO, y un nivel ALTO de salida se produce sólo cuando todas las entradas están a nivel BAJO, tenemos una puerta NOR.

FIGURA 15.47
Puerta PMOS básica.



NMOS

Los dispositivos **NMOS** se han desarrollado a medida que la tecnología de procesamiento mejoraba, y ahora la mayoría de las memorias y microprocesadores utilizan NMOS. En los circuitos NMOS se utiliza el transistor MOS de canal-*n*, como muestra la Figura 15.48 para una puerta NAND y una puerta NOR.

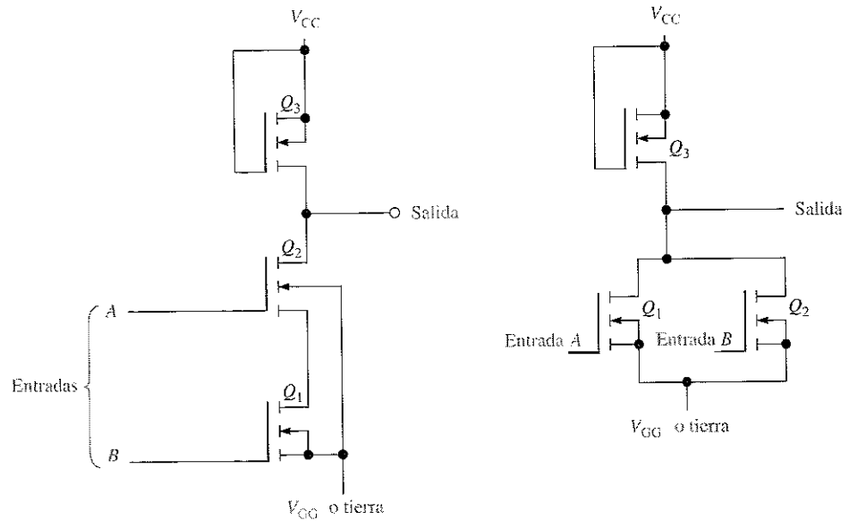
En la Figura 15.48(a), Q_3 actúa como una resistencia para limitar la corriente. Cuando se aplica un nivel BAJO (V_{GG} o tierra) a una o ambas entradas, entonces al menos uno de los transistores (Q_1 o Q_2) no conduce, y la salida se fuerza a un nivel ALTO próximo a V_{CC}. Cuando se aplican niveles ALTOS (V_{CC}) a ambas entradas A y B, los transistores Q_1 y Q_2 conducen, y la salida se pone a nivel BAJO. Por supuesto, este funcionamiento identifica al circuito como una puerta NAND.

En la Figura 15.48(b), Q_3 actúa de nuevo como resistencia. Un nivel ALTO en cualquier entrada hace que Q_1 o Q_2 conduzcan, forzando la salida a nivel BAJO. Cuando ambas entradas están a nivel BAJO, ningún transistor conduce y se fuerza la salida al nivel ALTO.

E²CMOS

La tecnología E²CMOS está basada en una combinación de las tecnologías CMOS y NMOS y se utiliza en los dispositivos GAL (Generic Array Logic) como los estudiados en los capítulos 7 y 11. Una celda E²CMOS se construye a partir de un transistor

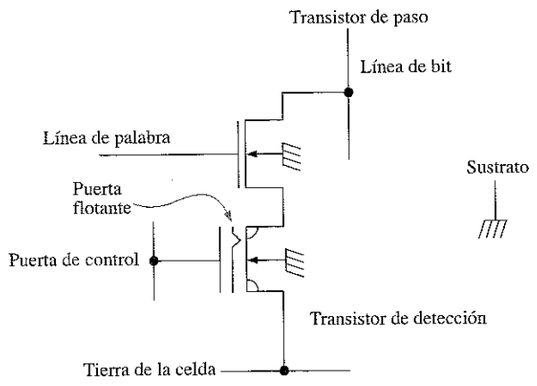
MOS con una puerta flotante, que se carga o descarga externamente por medio de una pequeña corriente de programación. La Figura 15.49 presenta un esquema de este tipo de celda.



(a) NAND (b) NOR

FIGURA 15.48
Dos puertas NMOS.

FIGURA 15.49
Celda E²CMOS.



Cuando la puerta flotante se carga a un potencial positivo eliminando electrones, el transistor de detección se activa, almacenando un cero binario. Cuando la puerta flotante se carga a un potencial negativo incorporando electrones, el transistor de detección se desactiva, almacenando un 1 binario. La puerta de control regula el potencial de la puerta flotante. El transistor de paso aísla de la matriz al transistor de detección durante las operaciones de lectura y de escritura, que utilizan las líneas de palabra y de bit.

La celda se programa aplicando un impulso de programación a la puerta de control o a la línea de bit de una celda, después de seleccionarla por medio de una tensión en la línea de palabra. Durante el ciclo de programación, en primer lugar, se borra la celda aplicando una tensión a la puerta de control para hacer negativa a la puerta flotante. Esto hace que el transistor de detección se quede en el estado de *bloqueo*, almacenando un 1. Para almacenar un 0 en la celda, se aplica un impulso de escritura a su línea de bit. Esto hará que la puerta flotante se cargue hasta un punto en el que el transistor de detección se activa, almacenando un 0. El bit almacenado en la celda se lee detectando la presencia o ausencia de una pequeña corriente de celda en la línea de bit. Cuando se almacena un 1, no hay corriente de celda, porque el transistor de detección está desactivado. Cuando se almacena un 0, existe una pequeña corriente de celda debido a que el transistor de detección está activado. Una vez que se ha almacenado el bit en la celda, permanecerá indefinidamente, a menos que se borre o se escriba un nuevo dato en ella.

REVISIÓN DE LA SECCIÓN 15.7

1. ¿Cuál es la característica principal de la tecnología NMOS y PMOS en circuitos integrados?
2. ¿Cuál es el mecanismo para almacenar la carga en una celda E²CMOS?

■ RESUMEN

■ *Fórmulas*

(15.1): $V_{NH} = V_{OH(mín)} - V_{IH(mín)}$ Margen de ruido para el nivel alto

(15.2): $V_{NL} = V_{IL(máx)} - V_{OL(máx)}$ Margen de ruido para el nivel bajo

(15.3): $I_{CC} = \dfrac{I_{CCH} + I_{CCL}}{2}$ Corriente continua media de alimentación

(15.4): $P_D = V_{CC} I_{CC}$ Disipación de potencia

(15.5): $R_p = \dfrac{V_{CC} - V_{OL(mín)}}{I_{OL(TTL)} + nI_{IL(CMOS)}}$ Resistencia de pull-up para la interfaz TTL-CMOS

- Las salidas totem-pole TTL no pueden conectarse juntas.
- Las salidas en colector abierto y en drenador abierto pueden conectarse en configuración AND-cableada.
- Los dispositivos CMOS ofrecen una disipación de potencia menor que cualquiera de la serie TTL.
- Un dispositivo TTL no es tan vulnerable a las descargas electrostáticas como un dispositivo CMOS.
- Debido a las descargas electrostáticas, los dispositivos CMOS deben manejarse con mucho cuidado.
- ECL es el tipo de circuito lógico más rápido.
- La tecnología E²CMOS se utiliza en los dispositivos GAL y otros dispositivos PLD.

■ PALABRAS CLAVE

Estos términos también se encuentran en el glosario del final del libro.

Bipolar Tecnología de transistores en la que un transistor se construye con dos uniones *pn*.

Carga unidad Una medida del fan-out. Una entrada de puerta representa una carga unidad para la salida de la puerta excitadora.

CMOS Complementary Metal-Oxide Semiconductor, metal-óxido semiconductor complementario. Un tipo de circuito de transistores que utiliza transistores MOSFET (metal-oxide semiconductor field-effect, transistor MOS de efecto de campo) de canal-*n* y canal-*p*.

Colector abierto Un tipo de salida en los circuitos TTL en el que el colector del transistor de salida se deja desconectado internamente y se encuentra disponible para conectarse a una carga externa que requiera una corriente o tensión relativamente altas.

Drenador Uno de los terminales de un MOSFET.

E²CMOS Electrically Erasable CMOS, CMOS eléctricamente borrable. Tecnología de circuitos utilizada en dispositivos lógicos programables (PLD).

Fuente Uno de los terminales de un MOSFET.

Fuente de corriente Funcionamiento de un circuito cuando su salida entrega corriente a la carga.

Inmunidad al ruido Capacidad de un circuito para rechazar señales no deseadas.

Margen de ruido La diferencia entre la salida máxima a nivel BAJO de una puerta y el máximo nivel de entrada a nivel BAJO aceptable por una puerta equivalente; también, diferencia entre la salida mínima a nivel ALTO de una puerta y el mínimo nivel de entrada a nivel ALTO aceptable por una puerta equivalente. El margen de ruido algunas veces se expresa como un porcentaje de la tensión de alimentación continua.

Puerta Circuito lógico que realiza una operación lógica específica, tal como AND u OR. Uno de los tres terminales de un transistor MOSFET.

Resistencia de pull-up Resistencia que se utiliza para mantener un determinado punto de un circuito a nivel ALTO cuando se encuentra en estado inactivo.

Retardo de propagación Intervalo de tiempo que transcurre entre la transición de entrada y su correspondiente transición de salida en un circuito lógico.

Sumidero de corriente Funcionamiento de un circuito cuando su salida acepta corriente de la carga.

TTL Transistor-Transistor Logic, lógica transistor-transistor: un tipo de circuito integrado que utiliza transistores de unión bipolares.

■ AUTOTEST

1. Cuando la frecuencia de la señal de entrada a una puerta CMOS aumenta, la disipación media de potencia
   (a) decrece **(b)** aumenta
   (c) no cambia **(d)** decrece exponencialmente
2. Los dispositivos CMOS son más fiables que los TTL en un entorno de alto ruido debido a su
   (a) menor margen de ruido **(b)** capacitancia de entrada
   (c) mayor margen de ruido **(d)** menor disipación de potencia
3. Es necesario manejar correctamente un dispositivo CMOS por su
   (a) frágil construcción **(b)** alta inmunidad al ruido
   (c) susceptibilidad a las descargas electrostáticas **(d)** baja disipación de potencia
4. ¿Cuál de los siguientes circuitos no es un circuito TTL?
   (a) 74F00 **(b)** 74AS00 **(c)** 74HC00 **(d)** 74ALS00
5. Una entrada en abierto de una puerta NOR TTL
   (a) actúa como un nivel BAJO **(b)** actúa como un nivel ALTO
   (c) debe conectarse a tierra **(d)** debería conectarse a V_{CC} a través de una resistencia
   (e) las respuestas (b) y (c) **(f)** las respuestas (a) y (c)
6. Una puerta TTL LS puede excitar un máximo de
   (a) 20 unidades de carga **(b)** diez unidades de carga
   (c) cuarenta unidades de carga **(d)** un número ilimitado de unidades de carga
7. Si dos entradas no utilizadas de una puerta TTL LS se conectan a una entrada que se está excitando con otra puerta TTL LS, el número total de unidades de carga que esa puerta puede excitar es
   (a) siete **(b)** ocho **(c)** diecisiete **(d)** ilimitado
8. La principal ventaja de ECL sobre TTL y CMOS es
   (a) ECL es más barata
   (b) ECL consume menos potencia
   (c) ECL está disponible en una mayor variedad de tipos de circuitos
   (d) ECL es más rápida
9. ECL no se puede utilizar en
   (a) entornos de alto ruido
   (b) entornos húmedos
   (c) aplicaciones de alta frecuencia
10. El mecanismo básico para almacenar un bit de datos en una celda E²CMOS es
    (a) la puerta de control **(b)** el drenador flotante
    (c) la puerta flotante **(d)** la corriente de celda

■ PROBLEMAS

SECCIÓN 15.1 Parámetros y características de operación básicas

1. Una determinada puerta lógica tiene un $V_{OH(mín)} = 2{,}2$ V, y excita a una puerta con un $V_{OH(mín)} = 2{,}5$ V. ¿Son compatibles estas puertas para el funcionamiento en estado ALTO? ¿Por qué?
2. Una determinada puerta lógica tiene un $V_{OL(máx)} = 0{,}45$ V y excita a una puerta con un $V_{IL(máx)} = 0{,}75$ V ¿Son compatibles estas puertas para el funcionamiento en estado BAJO? ¿Por qué?
3. Una puerta TTL tiene los siguiente valores de nivel de tensión: $V_{IH(mín)} = 2{,}25$ V, $V_{IL(máx)} = 0{,}65$ V. Suponer que la está excitando una puerta con $V_{OH(mín)} = 2{,}4$ V y $V_{OL(máx)} = 0{,}4$ V, ¿cuáles son los márgenes de ruido para los niveles ALTO y BAJO?

4. ¿Cuál es la máxima amplitud de los picos de ruido que se puede tolerar en las entradas de la puerta del problema 3, para el estado ALTO y el estado BAJO?

5. En la Tabla 15.3 se facilitan las especificaciones de tensión de tres tipos de puertas lógicas. Seleccionar la puerta que se debería utilizar en un entorno industrial de alto ruido.

TABLA 15.3

	$V_{OH(mín)}$	$V_{OL(máx)}$	$V_{IH(mín)}$	$V_{IL(máx)}$
Puerta *A*	2,4 V	0,4 V	2 V	0,8 V
Puerta *B*	3,5 V	0,2 V	2,5 V	0,6 V
Puerta *C*	4,2 V	0,2 V	3,2 V	0,8 V

6. Por una determinada puerta circula una corriente continua de alimentación procedente de una fuente de +5V y 2 mA en estado BAJO, y de 3,5 mA en estado ALTO. ¿Cuál es la disipación de potencia en el estado BAJO? ¿Cuál es la disipación de potencia en el estado ALTO? Suponiendo un ciclo de trabajo del 50%, ¿cuál es la disipación media de potencia?

7. Cada una de las puertas del circuito de la Figura 15.50 tiene un t_{PHL} y un t_{PLH} de 4 ns. Si se aplica a la entrada un impulso positivo como el indicado, ¿cuánto tiempo tardará en aparecer el impulso en la salida?

8. Para una determinada puerta, $t_{PLH} = 3$ ns y $t_{PHL} = 2$ ns. ¿Cuál es el retardo medio de propagación?

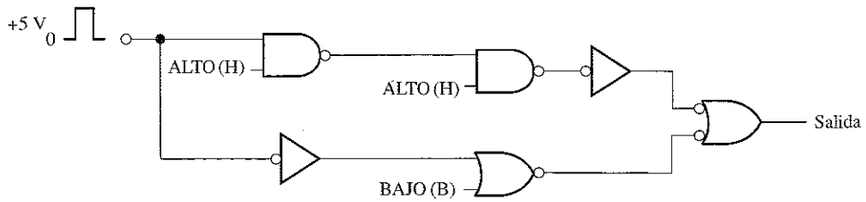


FIGURA 15.50

TABLA 15.4

	t_{PLH}	t_{PHL}	P_D
Puerta *A*	1 ns	1,2 ns	15 mW
Puerta *B*	5 ns	4 ns	8 mW
Puerta *C*	10 ns	10 ns	0,5 mW

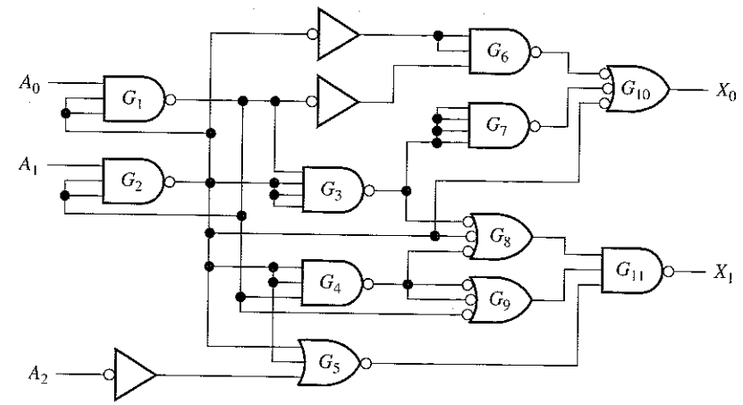
9. En la Tabla 15.4 se enumeran los parámetros de tres tipos de puertas. Basando su decisión en el producto velocidad-potencia, ¿cuál de ellas seleccionaría para obtener las máximas prestaciones?

10. ¿Qué puerta de la Tabla 15.4 se seleccionaría si se quiere que la puerta funcione a la frecuencia más alta posible?

11. Una puerta TTL estándar tiene un fan-out de 10. ¿Está sobrecargada alguna de las puertas de la Figura 15.51? Si la respuesta es afirmativa, ¿cuáles?

12. ¿Que red de puertas CMOS de la Figura 15.52 puede trabajar a la frecuencia más alta?

FIGURA 15.51



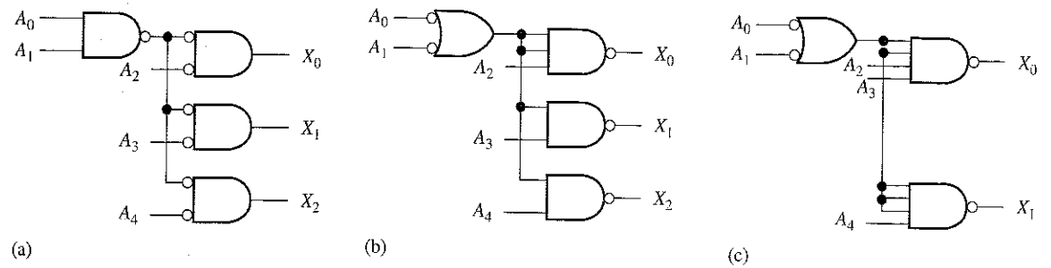


FIGURA 15.52

SECCIÓN 15.2 Circuitos CMOS

13. Determinar el estado (saturado o no, on u off) de cada MOSFET de la Figura 15.53.

14. La red de puertas CMOS de la Figura 15.54 está incompleta. Indicar los cambios que deberían hacerse.

15. Diseñar un circuito, utilizando las apropiadas puertas lógicas y/o inversores CMOS, en el que las señales procedentes de cuatro diferentes fuentes puedan conectarse a una línea común en diferentes instantes, sin interferir entre ellas.

FIGURA 15.53

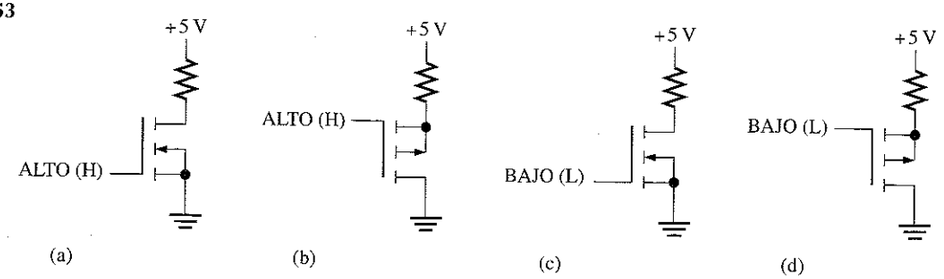
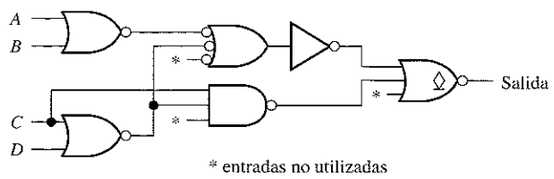


FIGURA 15.54



SECCIÓN 15.3 Circuitos TTL

16. Determinar cuáles BJT de la Figura 15.55 están saturados y cuáles no.

17. Determinar el estado de salida de cada una de las puertas TTL de la Figura 15.56.

18. La red de puertas TTL de la Figura 15.57 está incompleta. Indicar los cambios que habría que realizar.

FIGURA 15.55

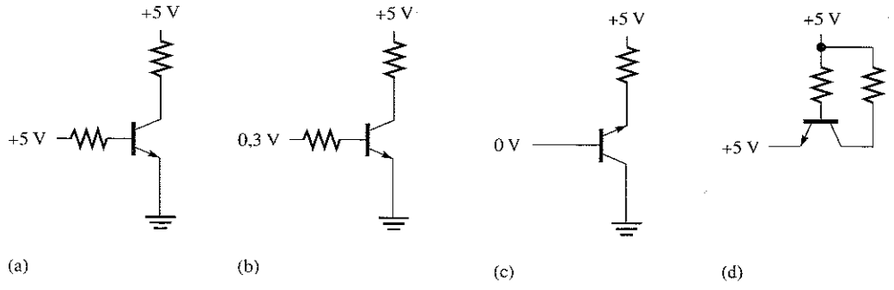


FIGURA 15.56

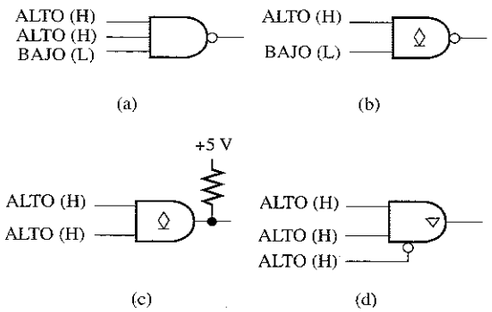
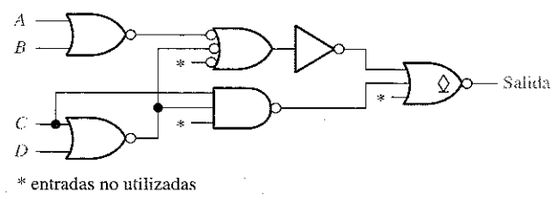


FIGURA 15.57



SECCIÓN 15.4 Consideraciones prácticas sobre el uso de TTL

19. Determinar el nivel de salida de cada puerta TTL de la Figura 15.58.

20. Para cada parte de la Figura 15.59, decir si la puerta excitadora actúa como fuente o sumidero de corriente. Especificar en cada caso la corriente máxima que sale o entra en la salida de la puerta o puertas excitadoras. Todas las puertas son TTL estándar.

FIGURA 15.58

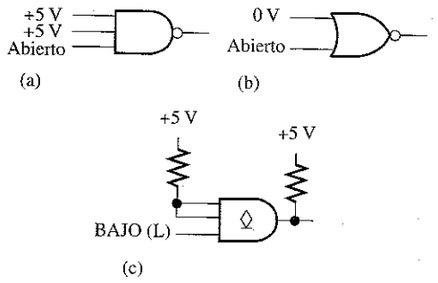
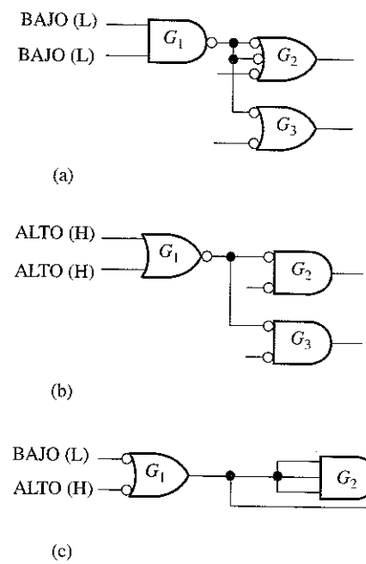


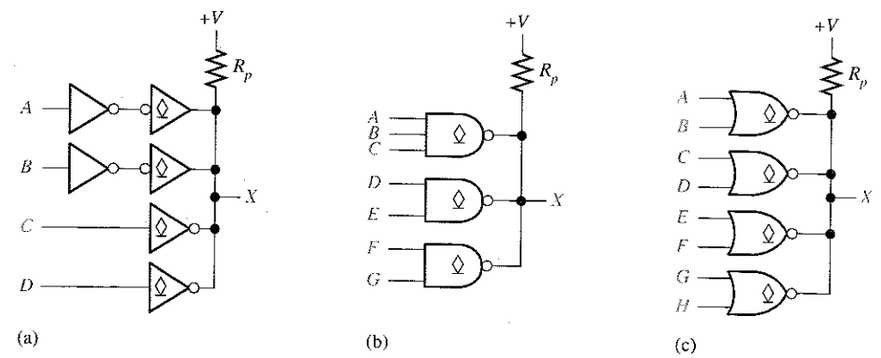
FIGURA 15.59



21. Utilizar inversores en colector abierto para implementar las siguientes expresiones lógicas:
(a) $X = \overline{A}\,\overline{B}\,\overline{C}$ **(b)** $X = A\overline{B}C\overline{D}$ **(c)** $X = ABC\overline{D}\,\overline{E}\,\overline{F}$

22. Escribir la expresión lógica para cada uno de los circuitos de la Figura 15.60.

FIGURA 15.60



23. Determinar el valor mínimo de la resistencia de pull-up de cada circuito de la Figura 15.60, si $I_{OL(máx)} = 40$ mA y $V_{OL(máx)} = 0{,}25$ V para cada puerta. Suponer que se excitan diez unidades de carga TTL estándar a partir de la salida X y que la tensión de alimentación es 5 V.

24. Un cierto relé requiere 60 mA. Diseñar una forma de excitar al relé utilizando puertas NAND en colector abierto con una $I_{OL(máx)} = 40$ mA.

SECCIÓN 15.5 Comparación de las prestaciones CMOS y TTL

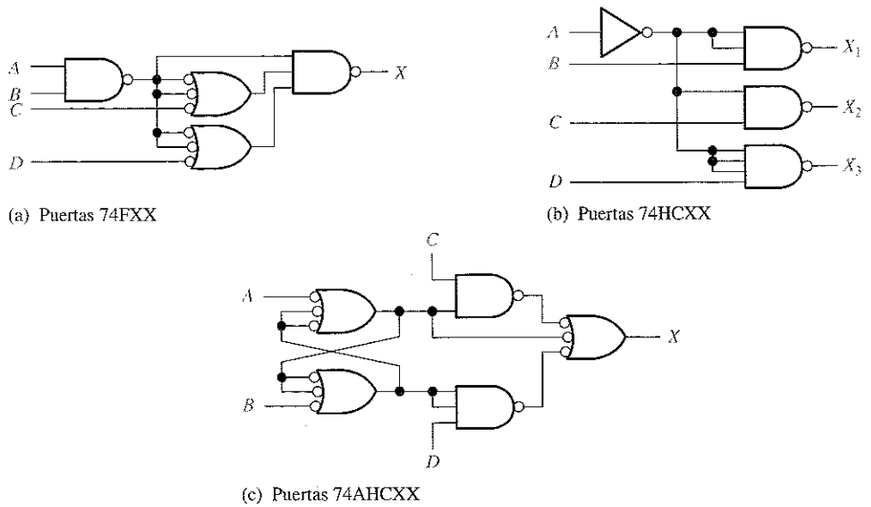
25. Seleccionar la familia de circuitos integrados con el mejor producto velocidad-potencia de la Tabla 15.1.

26. Determinar a partir de la Tabla 5.1 la familia lógica más apropiada para cada uno de los siguientes requisitos:

(a) retardo de propagación más corto
(b) velocidad de basculación de flip-flop más rápida
(c) disipación de potencia más baja
(d) mejor compromiso entre velocidad y potencia para una puerta lógica.

27. Determinar el retardo de propagación total desde cada entrada hasta cada salida para cada circuito de la Figura 15.61.

28. Uno de los flip-flops de la Figura 15.62 puede tener una salida errática, ¿cuál es y por qué?



(a) Puertas 74FXX

(b) Puertas 74HCXX

(c) Puertas 74AHCXX

FIGURA 15.61

SECCIÓN 15.6 Circuitos ECL

29. ¿Cuál es la diferencia básica entre los circuitos ECL y los circuitos TTL?

30. Elegir entre la tecnología ECL, la tecnología HCMOS o la apropiada serie TTL para cada uno de los siguientes requerimientos:

(a) muy alta velocidad
(b) muy baja potencia
(c) mejor compromiso entre alta velocidad y baja potencia

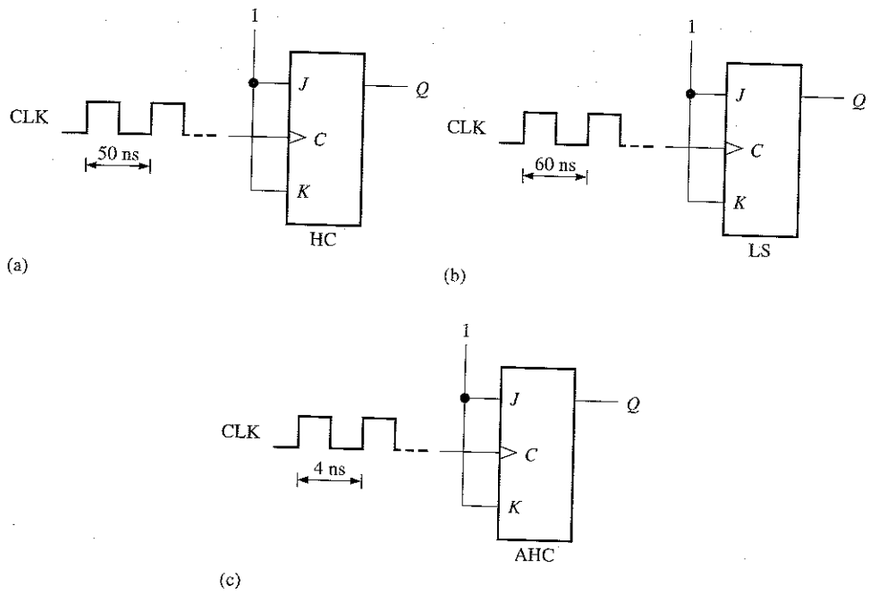


FIGURA 15.62

■ RESPUESTAS A LAS REVISIONES DE CADA SECCIÓN

SECCIÓN 15.1

1. V_{IH} : tensión de entrada a nivel ALTO. V_{IL} : tensión de entrada a nivel BAJO. V_{OH} : tensión de salida a nivel ALTO. V_{OL} : tensión de salida a nivel BAJO.
2. Es mejor un valor muy alto de margen de ruido.
3. La puerta B puede funcionar a una frecuencia mayor.
4. Una carga excesiva puede reducir el margen de ruido de una puerta.

SECCIÓN 15.2

1. Se utilizan transistores MOSFET en la lógica CMOS.
2. Un circuito con salida complementaria está formado por un MOSFET de canal-*n* y otro de canal-*p*.
3. Una descarga electrostática puede dañar los dispositivos CMOS.

SECCIÓN 15.3

1. Falso. El BJT *npn* no conduce.
2. Un BJT en el estado de saturación (on) es un interruptor cerrado y en el estado de no conducción (off) es un interruptor abierto.
3. Totem-pole y colector abierto son dos tipos de salida TTL.
4. La lógica de tres estados proporciona un estado de alta impedancia, en el que la salida está desconectada del resto del circuito.

SECCIÓN 15.4

1. Actúa como sumidero de corriente en el estado BAJO.

2. La corriente de fuente es menor que la corriente de sumidero porque una carga TTL es como un diodo polarizado en inversa en el estado ALTO.
3. Los transistores totem-pole no pueden soportar la corriente cuando una salida trata de adoptar un nivel ALTO y la otra un nivel BAJO.
4. La función AND-cableada debe utilizar salidas en colector abierto.
5. El excitador de lámpara debe tener una salida en colector abierto.
6. Falso, una entrada TTL no conectada generalmente actúa como un nivel ALTO.

SECCIÓN 15.5

1. BiCMOS utiliza transistores bipolares para los circuitos de entrada y salida y CMOS para los circuitos internos.
2. CMOS tiene una disipación de potencia más baja que la tecnología bipolar.

SECCIÓN 15.6

1. ECL es más rápida que TTL
2. ECL consume más y tiene menor margen de ruido que TTL.

SECCIÓN 15.7

1. NMOS y PMOS son de alta densidad
2. La puerta flotante es el mecanismo para almacenar la carga en una celda E^2CMOS.

■ RESPUESTAS A LOS PROBLEMAS RELACIONADOS DE LOS EJEMPLOS

15.1 CMOS

15.2 10,75 μW

15.3 $I_{T(\text{fuente})} = 5(20\ \mu\text{A}) = 100\ \mu\text{A}$
$I_{T(\text{sumidero})} = 5(-0{,}4\ \mu\text{A}) = -2{,}0\ \mu\text{A}$

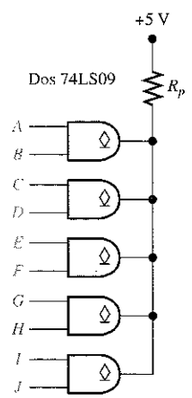
15.4 Fan-out = 20

15.5 $X = (\overline{AB})(\overline{CD})(\overline{EF})(\overline{GH}) = (\overline{A} + \overline{B})(\overline{C} + \overline{D})(\overline{E} + \overline{F})(\overline{G} + \overline{H})$

15.6. Véase la Figura 15.63.

15.7 $R_L = 97\ \Omega$

FIGURA 15.63



■ RESPUESTAS AL AUTOTEST

1. (b) **2.** (c) **3.** (c) **4.** (c) **5.** (e)
6. (a) **7.** (c) **8.** (d) **9.** (a) **10.** (c)

APÉNDICE A

HOJAS DE CARACTERÍSTICAS

SN5400, SN54LS00, SN54S00, SN7400, SN74LS00, SN74S00
QUADRUPLE 2-INPUT POSITIVE-NAND GATES

DECEMBER 1983 - REVISED MARCH 1988

- Package Options Include Plastic "Small Outline" Packages, Ceramic Chip Carriers and Flat Packages, and Plastic and Ceramic DIPs
- Dependable Texas Instruments Quality and Reliability

description

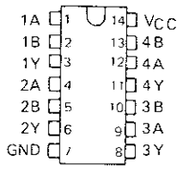
These devices contain four independent 2-input-NAND gates.

The SN5400, SN54LS00, and SN54S00 are characterized for operation over the full military temperature range of −55°C to 125°C. The SN7400, SN74LS00, and SN74S00 are characterized for operation from 0°C to 70°C.

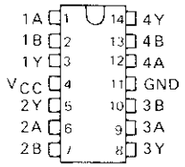
FUNCTION TABLE (each gate)

INPUTS		OUTPUT
A	B	Y
H	H	L
L	X	H
X	L	H

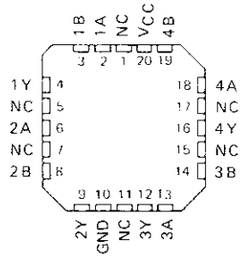
SN5400 . . . J PACKAGE
SN54LS00, SN54S00 . . . J OR W PACKAGE
SN7400 . . . N PACKAGE
SN74LS00, SN74S00 . . . D OR N PACKAGE
(TOP VIEW)



SN5400 . . . W PACKAGE
(TOP VIEW)

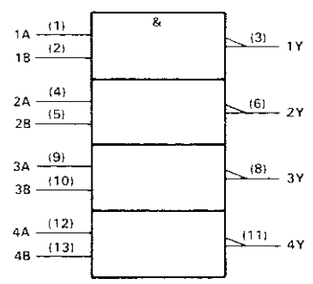


SN54LS00, SN54S00 . . . FK PACKAGE
(TOP VIEW)



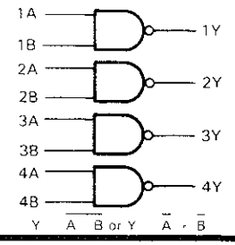
NC No internal connection

logic symbol†



†This symbol is in accordance with ANSI/IEEE Std. 91-1984 and IEC Publication 617-12.
Pin numbers shown are for D, J, and N packages.

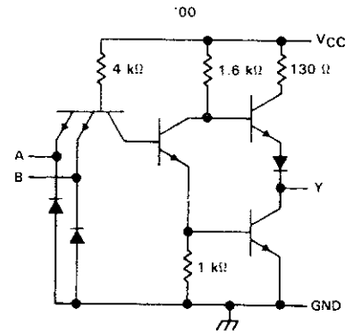
logic diagram (positive logic)



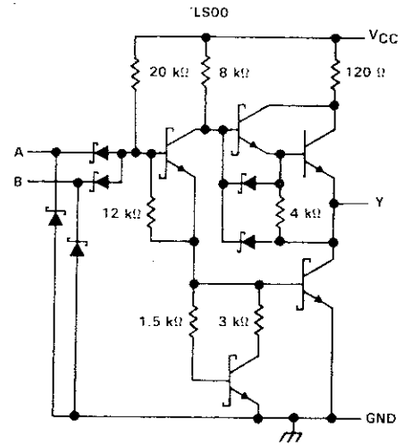
SN5400, SN54LS00, SN54S00, SN7400, SN74LS00, SN74S00
QUADRUPLE 2-INPUT POSITIVE-NAND GATES

schematics (each gate)

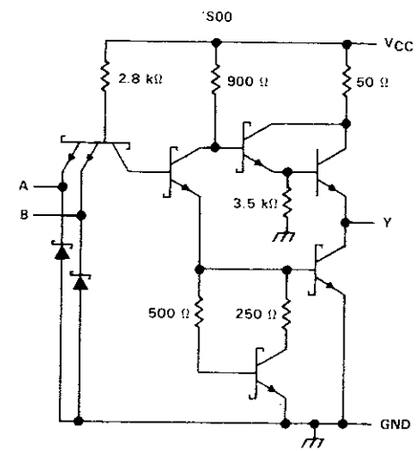
'00



'LS00



'S00



Resistor values shown are nominal.

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)

Supply voltage, V_{CC} (see Note 1) . . . . . 7 V
Input voltage: '00, 'S00 . . . . . 5.5 V
'LS00 . . . . . 7 V
Operating free-air temperature range: SN54' . . . . . −55°C to 125°C
SN74' . . . . . 0°C to 70°C
Storage temperature range . . . . . −65°C to 150°C

NOTE 1: Voltage values are with respect to network ground terminal.

SN5400, SN7400
QUADRUPLE 2-INPUT POSITIVE-NAND GATES

recommended operating conditions

		SN5400			SN7400			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
V_{CC}	Supply voltage	4.5	5	5.5	4.75	5	5.25	V
V_{IH}	High-level input voltage	2			2			V
V_{IL}	Low-level input voltage			0.8			0.8	V
I_{OH}	High-level output current			−0.4			−0.4	mA
I_{OL}	Low-level output current			16			16	mA
T_A	Operating free-air temperature	−55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER	TEST CONDITIONS †		SN5400			SN7400			UNIT
			MIN	TYP‡	MAX	MIN	TYP‡	MAX	
V_{IK}	V_{CC} = MIN,	I_I = −12 mA			−1.5			−1.5	V
V_{OH}	V_{CC} = MIN,	V_{IL} = 0.8 V, I_{OH} = −0.4 mA	2.4	3.4		2.4	3.4		V
V_{OL}	V_{CC} = MIN,	V_{IH} = 2 V, I_{OL} = 16 mA		0.2	0.4		0.2	0.4	V
I_I	V_{CC} = MAX,	V_I = 5.5 V			1			1	mA
I_{IH}	V_{CC} = MAX,	V_I = 2.4 V			40			40	μA
I_{IL}	V_{CC} = MAX,	V_I = 0.4 V			−1.6			−1.6	mA
I_{OS}§	V_{CC} = MAX		−20		−55	−18		−55	mA
I_{CCH}	V_{CC} = MAX,	V_I = 0 V		4	8		4	8	mA
I_{CCL}	V_{CC} = MAX,	V_I = 4.5 V		12	22		12	22	mA

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.
‡ All typical values are at V_{CC} = 5 V, T_A = 25°C.
§ Not more than one output should be shorted at a time.

switching characteristics, V_{CC} = 5 V, T_A = 25°C

PARAMETER	FROM (INPUT)	TO (OUTPUT)	TEST CONDITIONS		MIN	TYP	MAX	UNIT
t_{PLH}	A or B	Y	R_L = 400 Ω,	C_L = 15 pF		11	22	ns
t_{PHL}						7	15	ns

SN54LS00, SN74LS00
QUADRUPLE 2-INPUT POSITIVE-NAND GATES

recommended operating conditions

		SN54LS00			SN74LS00			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
V_{CC}	Supply voltage	4.5	5	5.5	4.75	5	5.25	V
V_{IH}	High-level input voltage	2			2			V
V_{IL}	Low-level input voltage			0.7			0.8	V
I_{OH}	High-level output current			−0.4			−0.4	mA
I_{OL}	Low-level output current			4			8	mA
T_A	Operating free-air temperature	−55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER	TEST CONDITIONS †		SN54LS00			SN74LS00			UNIT
			MIN	TYP‡	MAX	MIN	TYP‡	MAX	
V_{IK}	V_{CC} = MIN,	I_I = −18 mA			−1.5			−1.5	V
V_{OH}	V_{CC} = MIN,	V_{IL} = MAX, I_{OH} = −0.4 mA	2.5	3.4		2.7	3.4		V
V_{OL}	V_{CC} = MIN,	V_{IH} = 2 V, I_{OL} = 4 mA		0.25	0.4		0.25	0.4	V
	V_{CC} = MIN,	V_{IH} = 2 V, I_{OL} = 8 mA					0.35	0.5	
I_I	V_{CC} = MAX,	V_I = 7 V			0.1			0.1	mA
I_{IH}	V_{CC} = MAX,	V_I = 2.7 V			20			20	μA
I_{IL}	V_{CC} = MAX,	V_I = 0.4 V			−0.4			−0.4	mA
I_{OS}§	V_{CC} = MAX		−20		−100	−20		−100	mA
I_{CCH}	V_{CC} = MAX,	V_I = 0 V		0.8	1.6		0.8	1.6	mA
I_{CCL}	V_{CC} = MAX,	V_I = 4.5 V		2.4	4.4		2.4	4.4	mA

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.
‡ All typical values are at V_{CC} = 5 V, T_A = 25°C.
§ Not more than one output should be shorted at a time, and the duration of the short circuit should not exceed one second.

switching characteristics, V_{CC} = 5 V, T_A = 25°C

PARAMETER	FROM (INPUT)	TO (OUTPUT)	TEST CONDITIONS		MIN	TYP	MAX	UNIT
t_{PLH}	A or B	Y	R_L = 2 kΩ,	C_L = 15 pF		9	15	ns
t_{PHL}						10	15	ns

SN54F00, SN74F00
QUADRUPLE 2-INPUT POSITIVE-NAND GATES

SDFS035A – MARCH 1987 – REVISED OCTOBER 1993

- **Package Options Include Plastic Small-Outline Packages, Ceramic Chip Carriers, and Standard Plastic and Ceramic 300-mil DIPs**

description

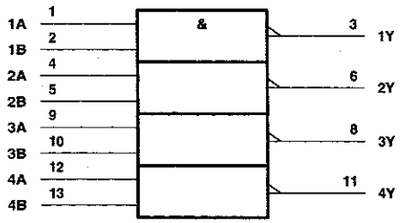
These devices contain four independent 2-input NAND gates. They perform the Boolean functions $Y = \overline{A \bullet B}$ or $Y = \overline{A} + \overline{B}$ in positive logic.

The SN54F00 is characterized for operation over the full military temperature range of −55°C to 125°C. The SN74F00 is characterized for operation from 0°C to 70°C.

FUNCTION TABLE
(each gate)

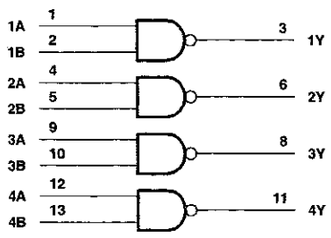
INPUTS		OUTPUT
A	B	Y
H	H	L
L	X	H
X	L	H

logic symbol†



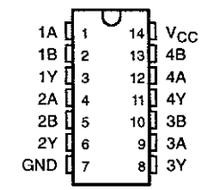
† This symbol is in accordance with ANSI/IEEE Std 91-1984 and IEC Publication 617-12.

logic diagram (positive logic)

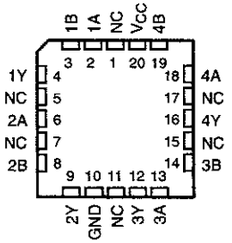


Pin numbers shown are for the D, J, and N packages.

SN54F00 . . . J PACKAGE
SN74F00 . . . D OR N PACKAGE
(TOP VIEW)



SN54F00 . . . FK PACKAGE
(TOP VIEW)



NC – No internal connection

SN54F00, SN74F00
QUADRUPLE 2-INPUT POSITIVE-NAND GATES

SDFS035A – MARCH 1987 – REVISED OCTOBER 1993

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Supply voltage range, V_{CC} ..... −0.5 V to 7 V
Input voltage range, V_I (see Note 1) ..... −1.2 V to 7 V
Input current range ..... −30 mA to 5 mA
Voltage range applied to any output in the high state ..... −0.5 V to V_{CC}
Current into any output in the low state ..... 40 mA
Operating free-air temperature range: SN54F00 ..... −55°C to 125°C
SN74F00 ..... 0°C to 70°C
Storage temperature range ..... −65°C to 150°C

† Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 1: The input voltage ratings may be exceeded provided the input current ratings are observed.

recommended operating conditions

		SN54F00			SN74F00			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
V_{CC}	Supply voltage	4.5	5	5.5	4.5	5	5.5	V
V_{IH}	High-level input voltage	2			2			V
V_{IL}	Low-level input voltage			0.8			0.8	V
I_{IK}	Input clamp current			−18			−18	mA
I_{OH}	High-level output current			−1			−1	mA
I_{OL}	Low-level output current			20			20	mA
T_A	Operating free-air temperature	−55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER	TEST CONDITIONS		SN54F00			SN74F00			UNIT
			MIN	TYP‡	MAX	MIN	TYP‡	MAX	
V_{IK}	V_{CC} = 4.5 V,	I_I = −18 mA			−1.2			−1.2	V
V_{OH}	V_{CC} = 4.5 V,	I_{OH} = −1 mA	2.5	3.4		2.5	3.4		V
	V_{CC} = 4.75 V,	I_{OH} = −1 mA				2.7			
V_{OL}	V_{CC} = 4.5 V,	I_{OL} = 20 mA		0.3	0.5		0.3	0.5	V
I_I	V_{CC} = 5.5 V,	V_I = 7 V			0.1			0.1	mA
I_{IH}	V_{CC} = 5.5 V,	V_I = 2.7 V			20			20	μA
I_{IL}	V_{CC} = 5.5 V,	V_I = 0.5 V			−0.6			−0.6	mA
I_{OS}§	V_{CC} = 5.5 V,	V_O = 0	−60		−150	−60		−150	mA
I_{CCH}	V_{CC} = 5.5 V,	V_I = 0		1.9	2.8		1.9	2.8	mA
I_{CCL}	V_{CC} = 5.5 V,	V_I = 4.5 V		6.8	10.2		6.8	10.2	mA

‡ All typical values are at V_{CC} = 5 V, T_A = 25°C.

§ Not more than one output should be shorted at a time, and the duration of the short circuit should not exceed one second.

SN54F00, SN74F00
QUADRUPLE 2-INPUT POSITIVE-NAND GATES

SDFS035A – MARCH 1987 – REVISED OCTOBER 1993

switching characteristics (see Note 2)

PARAMETER	FROM (INPUT)	TO (OUTPUT)	V_{CC} = 5 V, C_L = 50 pF, R_L = 500 Ω, T_A = 25°C			V_{CC} = 4.5 V to 5.5 V, C_L = 50 pF, R_L = 500 Ω, T_A = MIN to MAX†				UNIT
			'F00			SN54F00		SN74F00		
			MIN	TYP	MAX	MIN	MAX	MIN	MAX	
t_{PLH}	A or B	Y	1.6	3.3	5	2	7	1.6	6	ns
t_{PHL}			1	2.8	4.3	1.5	6.5	1	5.3	

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.
NOTE 2: Load circuits and waveforms are shown in Section 1.

TEXAS INSTRUMENTS

Data sheet acquired from Harris Semiconductor
SCHS117

August 1997

CD54HC04, CD54HCT04, CD74HC04, CD74HCT04

High Speed CMOS Logic Hex Inverter

Features

- Buffered Inputs
- Typical Propagation Delay: 6ns at V_{CC} = 5V, C_L = 15pF, T_A = 25°C
- Fanout (Over Temperature Range)
  - Standard Outputs . . . . . 10 LSTTL Loads
  - Bus Driver Outputs . . . . . 15 LSTTL Loads
- Wide Operating Temperature Range . . . -55°C to 125°C
- Balanced Propagation Delay and Transition Times
- Significant Power Reduction Compared to LSTTL Logic ICs
- HC Types
  - 2V to 6V Operation
  - High Noise Immunity: N_{IL} = 30%, N_{IH} = 30% of V_{CC} at V_{CC} = 5V
- HCT Types
  - 4.5V to 5.5V Operation
  - Direct LSTTL Input Logic Compatibility, V_{IL} = 0.8V (Max), V_{IH} = 2V (Min)
  - CMOS Input Compatibility, $I_I \leq 1\mu A$ at V_{OL}, V_{OH}

Description

The Harris CD54HC04, CD54HCT04, CD74HC04 and CD74HCT04 logic gates utilize silicon gate CMOS technology to achieve operating speeds similar to LSTTL gates with the low power consumption of standard CMOS integrated circuits. All devices have the ability to drive 10 LSTTL loads. The 74HCT logic family is functionally pin compatible with the standard 74LS logic family.

Ordering Information

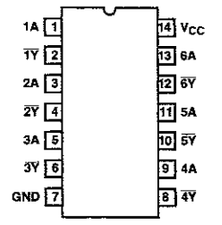
PART NUMBER	TEMP. RANGE (°C)	PACKAGE	PKG. NO.
CD74HC04E	-55 to 125	14 Ld PDIP	E14.3
CD74HCT04E	-55 to 125	14 Ld PDIP	E14.3
CD74HC04M	-55 to 125	14 Ld SOIC	M14.15
CD74HCT04M	-55 to 125	14 Ld SOIC	M14.15
CD54HC04F	-55 to 125	14 Ld CERDIP	F14.3
CD54HCT04F	-55 to 125	14 Ld CERDIP	F14.3
CD54HC04W	-55 to 125	Wafer	
CD54HCT04W	-55 to 125	Wafer	
CD54HC04H	-55 to 125	Die	
CD54HCT04H	-55 to 125	Die	

NOTE:

1. When ordering, use the entire part number. Add the suffix 96 to obtain the variant in the tape and reel.

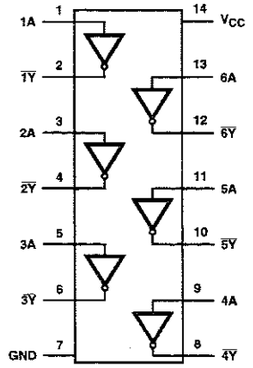
Pinout

CD54HC04, CD54HCT04, CD74HC04, CD74HCT04
(PDIP, CERDIP, SOIC)
TOP VIEW



CD54HC04, CD54HCT04, CD74HC04, CD74HCT04

Functional Diagram



TRUTH TABLE

INPUTS	
nA	nY
L	H
H	L

NOTE: H = High Voltage Level, L = Low Voltage Level

Logic Symbol

CD54HC04, CD54HCT04, CD74HC04, CD74HCT04

Absolute Maximum Ratings

DC Supply Voltage, V_{CC} . . . . . -0.5V to 7V
DC Input Diode Current, I_{IK}
For V_I < -0.5V or V_I > V_{CC} + 0.5V . . . . . ±20mA
DC Output Diode Current, I_{OK}
For V_O < -0.5V or V_O > V_{CC} + 0.5V . . . . . ±20mA
DC Output Source or Sink Current per Output Pin, I_O
For V_O > -0.5V or V_O < V_{CC} + 0.5V . . . . . ±25mA
DC V_{CC} or Ground Current, I_{CC} or I_{GND} . . . . . ±50mA

Operating Conditions

Temperature Range (T_A) . . . . . -55°C to 125°C
Supply Voltage Range, V_{CC}
HC Types . . . . . 2V to 6V
HCT Types . . . . . 4.5V to 5.5V
DC Input or Output Voltage, V_I, V_O . . . . . 0V to V_{CC}
Input Rise and Fall Time
2V . . . . . 1000ns (Max)
4.5V . . . . . 500ns (Max)
6V . . . . . 400ns (Max)

Thermal Information

Thermal Resistance (Typical, Note 2)	θ_{JA} (°C/W)	θ_{JC} (°C/W)
PDIP Package	100	N/A
CERDIP Package	130	55
SOIC Package	180	N/A

Maximum Junction Temperature (Hermetic Package or Die) . . . 175°C
Maximum Junction Temperature (Plastic Package) . . . . . 150°C
Maximum Storage Temperature Range . . . . . -65°C to 150°C
Maximum Lead Temperature (Soldering 10s) . . . . . 300°C
(SOIC - Lead Tips Only)

CAUTION: Stresses above those listed in "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

NOTE:

2. θ_{JA} is measured with the component mounted on an evaluation PC board in free air.

DC Electrical Specifications

PARAMETER	SYMBOL	TEST CONDITIONS V_I (V)	TEST CONDITIONS I_O (mA)	V_{CC} (V)	25°C MIN	25°C TYP	25°C MAX	-40°C TO +85°C MIN	-40°C TO +85°C MAX	-55°C TO 125°C MIN	-55°C TO 125°C MAX	UNITS
HC TYPES												
High Level Input Voltage	V_{IH}	-	-	2	1.5	-	-	1.5	-	1.5	-	V
				4.5	3.15	-	-	3.15	-	3.15	-	V
				6	4.2	-	-	4.2	-	4.2	-	V
Low Level Input Voltage	V_{IL}	-	-	2	-	-	0.5	-	0.5	-	0.5	V
				4.5	-	-	1.35	-	1.35	-	1.35	V
				6	-	-	1.8	-	1.8	-	1.8	V
High Level Output Voltage CMOS Loads	V_{OH}	V_{IH} or V_{IL}	-0.02	2	1.9	-	-	1.9	-	1.9	-	V
			-0.02	4.5	4.4	-	-	4.4	-	4.4	-	V
			-0.02	6	5.9	-	-	5.9	-	5.9	-	V
High Level Output Voltage TTL Loads			-	-	-	-	-	-	-	-	-	V
			-4	4.5	3.98	-	-	3.84	-	3.7	-	V
			-5.2	6	5.48	-	-	5.34	-	5.2	-	V
Low Level Output Voltage CMOS Loads	V_{OL}	V_{IH} or V_{IL}	0.02	2	-	-	0.1	-	0.1	-	0.1	V
			0.02	4.5	-	-	0.1	-	0.1	-	0.1	V
			0.02	6	-	-	0.1	-	0.1	-	0.1	V
Low Level Output Voltage TTL Loads			-	-	-	-	-	-	-	-	-	V
			4	4.5	-	-	0.26	-	0.33	-	0.4	V
			5.2	6	-	-	0.26	-	0.33	-	0.4	V
Input Leakage Current	I_I	V_{CC} or GND	-	6	-	-	±0.1	-	±1	-	±1	µA

CD54HC04, CD54HCT04, CD74HC04, CD74HCT04

DC Electrical Specifications (Continued)

PARAMETER	SYMBOL	TEST CONDITIONS V_I (V)	TEST CONDITIONS I_O (mA)	V_{CC} (V)	25°C MIN	25°C TYP	25°C MAX	-40°C TO +85°C MIN	-40°C TO +85°C MAX	-55°C TO 125°C MIN	-55°C TO 125°C MAX	UNITS
Quiescent Device Current	I_{CC}	V_{CC} or GND	0	6	-	-	2	-	20	-	40	µA
HCT TYPES												
High Level Input Voltage	V_{IH}	-	-	4.5 to 5.5	2	-	-	2	-	2	-	V
Low Level Input Voltage	V_{IL}	-	-	4.5 to 5.5	-	-	0.8	-	0.8	-	0.8	V
High Level Output Voltage CMOS Loads	V_{OH}	V_{IH} or V_{IL}	-0.02	4.5	4.4	-	-	4.4	-	4.4	-	V
High Level Output Voltage TTL Loads			-4	4.5	3.98	-	-	3.84	-	3.7	-	V
Low Level Output Voltage CMOS Loads	V_{OL}	V_{IH} or V_{IL}	0.02	4.5	-	-	0.1	-	0.1	-	0.1	V
Low Level Output Voltage TTL Loads			4	4.5	-	-	0.26	-	0.33	-	0.4	V
Input Leakage Current	I_I	V_{CC} and GND	0	5.5	-		±0.1	-	±1	-	±1	µA
Quiescent Device Current	I_{CC}	V_{CC} or GND	0	5.5	-	-	2	-	20	-	40	µA
Additional Quiescent Device Current Per Input Pin: 1 Unit Load (Note)	ΔI_{CC}	V_{CC} - 2.1	-	4.5 to 5.5	-	100	360	-	450	-	490	µA

NOTE: For dual-supply systems theoretical worst case (V_I = 2.4V, V_{CC} = 5.5V) specification is 1.8mA.

HCT Input Loading Table

INPUT	UNIT LOADS
nB	1.2

NOTE: Unit Load is ΔI_{CC} limit specified in DC Electrical Specifications table, e.g. 360µA max at 25°C.

Switching Specifications Input t_r, t_f = 6ns

PARAMETER	SYMBOL	TEST CONDITIONS	V_{CC} (V)	25°C MIN	25°C TYP	25°C MAX	-40°C TO 85°C MIN	-40°C TO 85°C MAX	-55°C TO 125°C MIN	-55°C TO 125°C MAX	UNITS
HC TYPES											
Propagation Delay, Input to Output (Figure 1)	t_{PLH}, t_{PHL}	C_L = 50pF	2	-	-	85	-	105	-	130	ns
			4.5	-	-	7	-	21	-	67	ns
			6	-	-	14	-	18	-	22	ns
Propagation Delay, Data Input to Output Y	t_{PLH}, t_{PHL}	C_L = 15pF	5	-	6	-	-	-	-	-	ns

CD54HC04, CD54HCT04, CD74HC04, CD74HCT04

Switching Specifications Input t_r, t_f = 6ns (Continued)

PARAMETER	SYMBOL	TEST CONDITIONS	V_{CC} (V)	25°C MIN	25°C TYP	25°C MAX	-40°C TO 85°C MIN	-40°C TO 85°C MAX	-55°C TO 125°C MIN	-55°C TO 125°C MAX	UNITS
Transition Times (Figure 1)	t_{TLH}, t_{THL}	C_L = 50pF	2	-	-	75	-	95	18	110	ns
			4.5	-	-	15	-	19	-	22	ns
			6	-	-	13	-	16	-	19	ns
Input Capacitance	C_I	-	-	-	-	10	-	10	-	10	pF
Power Dissipation Capacitance (Notes 3, 4)	C_{PD}	-	5	-	21	-	-	-	-	-	pF
HCT TYPES											
Propagation Delay, Input to Output (Figure 2)	t_{PLH}, t_{PHL}	C_L = 50pF	4.5	-	-	19	-	24	-	29	ns
Propagation Delay, Data Input to Output Y	t_{PLH}, t_{PHL}	C_L = 15pF	5	-	7	-	-	-	-	-	ns
Transition Times (Figure 2)	t_{TLH}, t_{THL}	C_L = 50pF	4.5	-	-	15	-	19	-	22	ns
Input Capacitance	C_I	-	-	-	-	10	-	10	-	10	pF
Power Dissipation Capacitance (Notes 3, 4)	C_{PD}	-	5	-	24	-	-	-	-	-	pF

NOTES:

3. C_{PD} is used to determine the dynamic power consumption, per gate.
4. $P_D = V_{CC}^2 f_i (C_{PD} + C_L)$ where f_i = input frequency, C_L = output load capacitance, V_{CC} = supply voltage.

Test Circuits and Waveforms

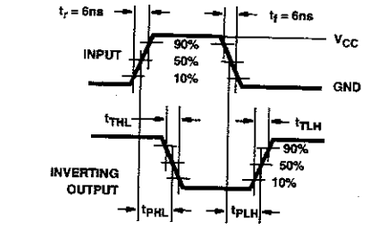


FIGURE 1. HC TRANSITION TIMES AND PROPAGATION DELAY TIMES, COMBINATION LOGIC

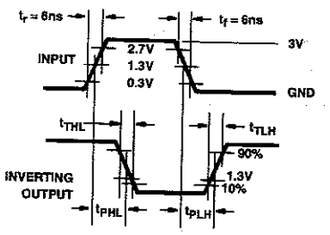


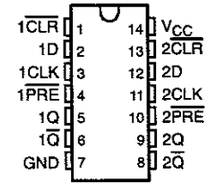
FIGURE 2. HCT TRANSITION TIMES AND PROPAGATION DELAY TIMES, COMBINATION LOGIC

SN54AHC74, SN74AHC74
DUAL POSITIVE-EDGE-TRIGGERED D-TYPE FLIP-FLOPS WITH CLEAR AND PRESET

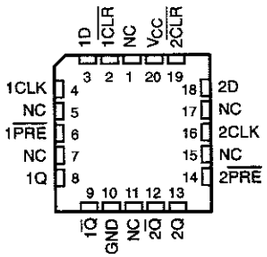
SCLS255E – DECEMBER 1995 – REVISED NOVEMBER 1998

- **Operating Range 2-V to 5.5-V V_{CC}**
- ***EPIC*™ (Enhanced-Performance Implanted CMOS) Process**
- **Latch-Up Performance Exceeds 250 mA Per JESD 17**
- **ESD Protection Exceeds 2000 V Per MIL-STD-883, Method 3015; Exceeds 200 V Using Machine Model (C = 200 pF, R = 0)**
- **Package Options Include Plastic Small-Outline (D), Shrink Small-Outline (DB), Thin Very Small-Outline (DGV), Thin Shrink Small-Outline (PW), and Ceramic Flat (W) Packages, Ceramic Chip Carriers (FK), and Standard Plastic (N) and Ceramic (J) DIPs**

SN54AHC74 . . . J OR W PACKAGE
SN74AHC74 . . . D, DB, DGV, N, OR PW PACKAGE
(TOP VIEW)



SN54AHC74 . . . FK PACKAGE
(TOP VIEW)



NC – No internal connection

description

The 'AHC74 devices are dual positive-edge-triggered D-type flip-flops.

A low level at the preset ($\overline{PRE}$) or clear ($\overline{CLR}$) inputs sets or resets the outputs, regardless of the levels of the other inputs. When $\overline{PRE}$ and $\overline{CLR}$ are inactive (high), data at the data (D) input meeting the setup time requirements is transferred to the outputs on the positive-going edge of the clock pulse. Clock triggering occurs at a voltage level and is not directly related to the rise time of the clock pulse. Following the hold-time interval, data at the D input can be changed without affecting the levels at the outputs.

The SN54AHC74 is characterized for operation over the full military temperature range of –55°C to 125°C. The SN74AHC74 is characterized for operation from –40°C to 85°C.

FUNCTION TABLE

INPUTS				OUTPUTS	
$\overline{PRE}$	$\overline{CLR}$	CLK	D	Q	$\overline{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H†	H†
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q_0	$\overline{Q}_0$

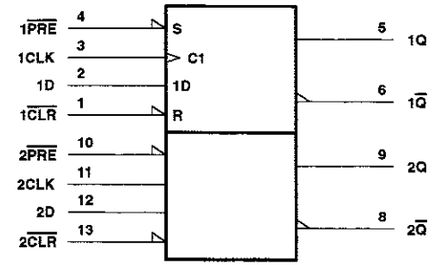
† This configuration is nonstable; that is, it does not persist when $\overline{PRE}$ or $\overline{CLR}$ returns to its inactive (high) level.

Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

EPIC is a trademark of Texas Instruments Incorporated.

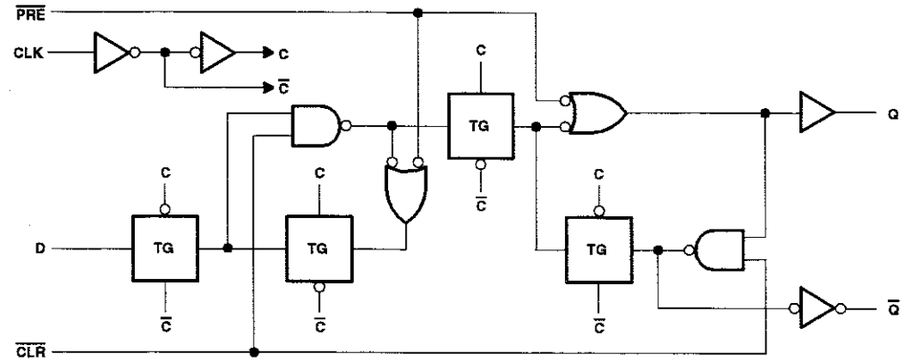
SN54AHC74, SN74AHC74
DUAL POSITIVE-EDGE-TRIGGERED D-TYPE FLIP-FLOPS WITH CLEAR AND PRESET
SCLS255E – DECEMBER 1995 – REVISED NOVEMBER 1998

logic symbol†



† This symbol is in accordance with ANSI/IEEE Std 91-1984 and IEC Publication 617-12.
Pin numbers shown are for the D, DB, DGV, J, N, PW, and W packages.

logic diagram, each flip-flop (positive logic)



absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Supply voltage range, V_{CC} ..... –0.5 V to 7 V
Input voltage range, V_I (see Note 1) ..... –0.5 V to 7 V
Output voltage range, V_O (see Note 1) ..... –0.5 V to V_{CC} + 0.5 V
Input clamp current, I_{IK} (V_I < 0) ..... –20 mA
Output clamp current, I_{OK} (V_O < 0 or V_O > V_{CC}) ..... ±20 mA
Continuous output current, I_O (V_O = 0 to V_{CC}) ..... ±25 mA
Continuous current through V_{CC} or GND ..... ±50 mA
Package thermal impedance, θ_{JA} (see Note 2): D package ..... 127°C/W
DB package ..... 158°C/W
DGV package ..... 182°C/W
N package ..... 78°C/W
PW package ..... 170°C/W
Storage temperature range, T_{stg} ..... –65°C to 150°C

† Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTES: 1. The input and output voltage ratings may be exceeded if the input and output current ratings are observed.
2. The package thermal impedance is calculated in accordance with JESD 51, except for through-hole packages, which use a trace length of zero.

recommended operating conditions (see Note 3)

			SN54AHC74		SN74AHC74		UNIT
			MIN	MAX	MIN	MAX	
V_{CC}	Supply voltage		2	5.5	2	5.5	V
V_{IH}	High-level input voltage	V_{CC} = 2 V	1.5		1.5		V
		V_{CC} = 3 V	2.1		2.1		
		V_{CC} = 5.5 V	3.85		3.85		
V_{IL}	Low-level input voltage	V_{CC} = 2 V		0.5		0.5	V
		V_{CC} = 3 V		0.9		0.9	
		V_{CC} = 5.5 V		1.65		1.65	
V_I	Input voltage		0	5.5	0	5.5	V
V_O	Output voltage		0	V_{CC}	0	V_{CC}	V
I_{OH}	High-level output current	V_{CC} = 2 V		–50		–50	μA
		V_{CC} = 3.3 V ± 0.3 V		–4		–4	mA
		V_{CC} = 5 V ± 0.5 V		–8		–8	
I_{OL}	Low-level output current	V_{CC} = 2 V		50		50	μA
		V_{CC} = 3.3 V ± 0.3 V		4		4	mA
		V_{CC} = 5 V ± 0.5 V		8		8	
$\Delta t/\Delta v$	Input transition rise or fall rate	V_{CC} = 3.3 V ± 0.3 V		100		100	ns/V
		V_{CC} = 5 V ± 0.5 V		20		20	
T_A	Operating free-air temperature		–55	125	–40	85	°C

NOTE 3: All unused inputs of the device must be held at V_{CC} or GND to ensure proper device operation. Refer to the TI application report, *Implications of Slow or Floating CMOS Inputs*, literature number SCBA004.

SN54AHC74, SN74AHC74
DUAL POSITIVE-EDGE-TRIGGERED D-TYPE FLIP-FLOPS
WITH CLEAR AND PRESET
SCLS255E – DECEMBER 1995 – REVISED NOVEMBER 1998

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS	V_{CC}	T_A = 25°C			SN54AHC74		SN74AHC74		UNIT
				MIN	TYP	MAX	MIN	MAX	MIN	MAX	
V_{OH}		I_{OH} = –50 μA	2 V	1.9	2		1.9		1.9		V
			3 V	2.9	3		2.9		2.9		
			4.5 V	4.4	4.5		4.4		4.4		
		I_{OH} = –4 mA	3 V	2.58			2.48		2.48		
		I_{OH} = –8 mA	4.5 V	3.94			3.8		3.8		
V_{OL}		I_{OL} = 50 μA	2 V			0.1		0.1		0.1	V
			3 V			0.1		0.1		0.1	
			4.5 V			0.1		0.1		0.1	
		I_{OL} = 4 mA	3 V			0.36		0.5		0.44	
		I_{OL} = 8 mA	4.5 V			0.36		0.5		0.44	
I_I	Data inputs	V_I = V_{CC} or GND	5.5 V			±0.1		±1		±1	μA
	Control inputs					±0.1		±1		±1	
I_{CC}		V_I = V_{CC} or GND, I_O = 0	5.5 V			2		20		20	μA
C_i		V_I = V_{CC} or GND	5 V		2	10				10	pF

timing requirements over recommended operating free-air temperature range, V_{CC} = 3.3 V ± 0.3 V (unless otherwise noted) (see Figure 1)

			T_A = 25°C		SN54AHC74		SN74AHC74		UNIT
			MIN	MAX	MIN	MAX	MIN	MAX	
t_w	Pulse duration	PRE or CLR low	6		7		7		ns
		CLK	6		7		7		
t_{su}	Setup time before CLK↑	Data	6		7		7		ns
		PRE or CLR inactive	5		5		5		
t_h	Hold time, data after CLK↑		0.5		0.5		0.5		ns

timing requirements over recommended operating free-air temperature range, V_{CC} = 5 V ± 0.5 V (unless otherwise noted) (see Figure 1)

			T_A = 25°C		SN54AHC74		SN74AHC74		UNIT
			MIN	MAX	MIN	MAX	MIN	MAX	
t_w	Pulse duration	PRE or CLR low	5		5		5		ns
		CLK	5		5		5		
t_{su}	Setup time before CLK↑	Data	5		5		5		ns
		PRE or CLR inactive	3		3		3		
t_h	Hold time, data after CLK↑		0.5		0.5		0.5		ns

switching characteristics over recommended operating free-air temperature range, V_{CC} = 3.3 V ± 0.3 V (unless otherwise noted) (see Figure 1)

PARAMETER	FROM (INPUT)	TO (OUTPUT)	LOAD CAPACITANCE	SN54AHC74					UNIT
				T_A = 25°C			MIN	MAX	
				MIN	TYP	MAX			
f_{max}			C_L = 15 pF*	80	125		70		MHz
			C_L = 50 pF	50	75		45		
t_{PLH}*	$\overline{PRE}$ or $\overline{CLR}$	Q or $\overline{Q}$	C_L = 15 pF		7.6	12.3	1	14.5	ns
t_{PHL}*					7.6	12.3	1	14.5	
t_{PLH}*	CLK	Q or $\overline{Q}$	C_L = 15 pF		6.7	11.9	1	14	ns
t_{PHL}*					6.7	11.9	1	14	
t_{PLH}	$\overline{PRE}$ or $\overline{CLR}$	Q or $\overline{Q}$	C_L = 50 pF		10.1	15.8	1	18	ns
t_{PHL}					10.1	15.8	1	18	
t_{PLH}	CLK	Q or $\overline{Q}$	C_L = 50 pF		9.2	15.4	1	17.5	ns
t_{PHL}					9.2	15.4	1	17.5	

* On products compliant to MIL-PRF-38535, this parameter is not production tested.

switching characteristics over recommended operating free-air temperature range, V_{CC} = 3.3 V ± 0.3 V (unless otherwise noted) (see Figure 1)

PARAMETER	FROM (INPUT)	TO (OUTPUT)	LOAD CAPACITANCE	SN74AHC74					UNIT
				T_A = 25°C			MIN	MAX	
				MIN	TYP	MAX			
f_{max}			C_L = 15 pF	80	125		70		MHz
			C_L = 50 pF	50	75		45		
t_{PLH}	$\overline{PRE}$ or $\overline{CLR}$	Q or $\overline{Q}$	C_L = 15 pF		7.6	12.3	1	14.5	ns
t_{PHL}					7.6	12.3	1	14.5	
t_{PLH}	CLK	Q or $\overline{Q}$	C_L = 15 pF		6.7	11.9	1	14	ns
t_{PHL}					6.7	11.9	1	14	
t_{PLH}	$\overline{PRE}$ or $\overline{CLR}$	Q or $\overline{Q}$	C_L = 50 pF		10.1	15.8	1	18	ns
t_{PHL}					10.1	15.8	1	18	
t_{PLH}	CLK	Q or $\overline{Q}$	C_L = 50 pF		9.2	15.4	1	17.5	ns
t_{PHL}					9.2	15.4	1	17.5	

SN54AHC74, SN74AHC74
DUAL POSITIVE-EDGE-TRIGGERED D-TYPE FLIP-FLOPS
WITH CLEAR AND PRESET
SCLS255E – DECEMBER 1995 – REVISED NOVEMBER 1998

switching characteristics over recommended operating free-air temperature range, V_{CC} = 5 V ± 0.5 V (unless otherwise noted) (see Figure 1)

PARAMETER	FROM (INPUT)	TO (OUTPUT)	LOAD CAPACITANCE	SN54AHC74					UNIT
				T_A = 25°C			MIN	MAX	
				MIN	TYP	MAX			
f_{max}			C_L = 15 pF*	130	170		110		MHz
			C_L = 50 pF	90	115		75		
t_{PLH}*	$\overline{PRE}$ or $\overline{CLR}$	Q or $\overline{Q}$	C_L = 15 pF		4.8	7.7	1	9	ns
t_{PHL}*					4.8	7.7	1	9	
t_{PLH}*	CLK	Q or $\overline{Q}$	C_L = 15 pF		4.6	7.3	1	8.5	ns
t_{PHL}*					4.6	7.3	1	8.5	
t_{PLH}	$\overline{PRE}$ or $\overline{CLR}$	Q or $\overline{Q}$	C_L = 50 pF		6.3	9.7	1	11	ns
t_{PHL}					6.3	9.7	1	11	
t_{PLH}	CLK	Q or Q	C_L = 50 pF		6.1	9.3	1	10.5	ns
t_{PHL}					6.1	9.3	1	10.5	

* On products compliant to MIL-PRF-38535, this parameter is not production tested.

switching characteristics over recommended operating free-air temperature range, V_{CC} = 5 V ± 0.5 V (unless otherwise noted) (see Figure 1)

PARAMETER	FROM (INPUT)	TO (OUTPUT)	LOAD CAPACITANCE	SN74AHC74					UNIT
				T_A = 25°C			MIN	MAX	
				MIN	TYP	MAX			
f_{max}			C_L = 15 pF	130	170		110		MHz
			C_L = 50 pF	90	115		75		
t_{PLH}	$\overline{PRE}$ or $\overline{CLR}$	Q or $\overline{Q}$	C_L = 15 pF		4.8	7.7	1	9	ns
t_{PHL}					4.8	7.7	1	9	
t_{PLH}	CLK	Q or $\overline{Q}$	C_L = 15 pF		4.6	7.3	1	8.5	ns
t_{PHL}					4.6	7.3	1	8.5	
t_{PLH}	$\overline{PRE}$ or $\overline{CLR}$	Q or $\overline{Q}$	C_L = 50 pF		6.3	9.7	1	11	ns
t_{PHL}					6.3	9.7	1	11	
t_{PLH}	CLK	Q or Q	C_L = 50 pF		6.1	9.3	1	10.5	ns
t_{PHL}					6.1	9.3	1	10.5	

noise characteristics, V_{CC} = 5 V, C_L = 50 pF, T_A = 25°C (see Note 4)

PARAMETER		SN74AHC74		UNIT
		MIN	MAX	
$V_{OL(P)}$	Quiet output, maximum dynamic V_{OL}		0.8	V
$V_{OL(V)}$	Quiet output, minimum dynamic V_{OL}		–0.8	V
$V_{OH(V)}$	Quiet output, minimum dynamic V_{OH}	4.7		V
$V_{IH(D)}$	High-level dynamic input voltage	3.5		V
$V_{IL(D)}$	Low-level dynamic input voltage		1.5	V

NOTE 4: Characteristics are for surface-mount packages only.

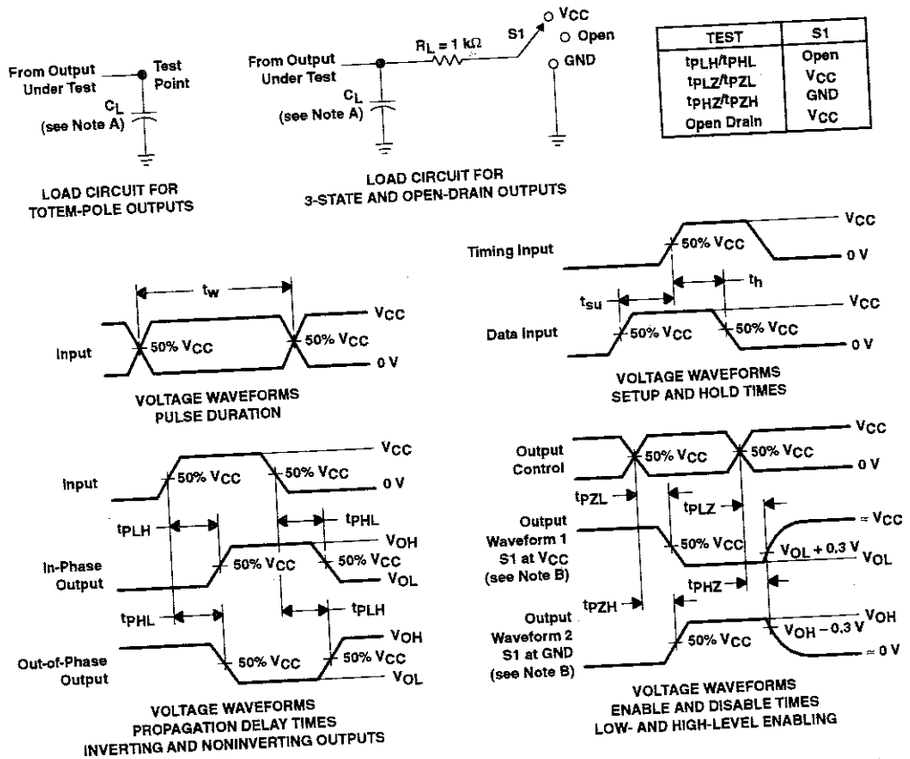
operating characteristics, V_{CC} = 5 V, T_A = 25°C

PARAMETER		TEST CONDITIONS		TYP	UNIT
C_{pd}	Power dissipation capacitance	No load,	f = 1 MHz	32	pF

SN54AHC74, SN74AHC74
DUAL POSITIVE-EDGE-TRIGGERED D-TYPE FLIP-FLOPS WITH CLEAR AND PRESET

SCLS255E – DECEMBER 1995 – REVISED NOVEMBER 1998

PARAMETER MEASUREMENT INFORMATION



NOTES: A. C_L includes probe and jig capacitance.
B. Waveform 1 is for an output with internal conditions such that the output is low except when disabled by the output control. Waveform 2 is for an output with internal conditions such that the output is high except when disabled by the output control.
C. All input pulses are supplied by generators having the following characteristics: PRR ≤ 1 MHz, Z_O = 50 Ω, t_r ≤ 3 ns, t_f ≤ 3 ns.
D. The outputs are measured one at a time with one input transition per measurement.

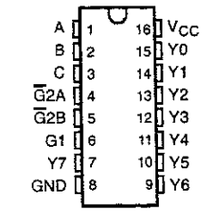
Figure 1. Load Circuit and Voltage Waveforms

SN54HC138, SN74HC138
3-LINE TO 8-LINE DECODERS/DEMULTIPLEXERS

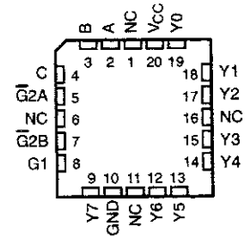
SCLS107C – DECEMBER 1982 – REVISED MAY 1997

- **Designed Specifically for High-Speed Memory Decoders and Data Transmission Systems**
- **Incorporate Three Enable Inputs to Simplify Cascading and/or Data Reception**
- **Package Options Include Plastic Small-Outline (D), Thin Shrink Small-Outline (PW), and Ceramic Flat (W) Packages, Ceramic Chip Carriers (FK), and Standard Plastic (N) and Ceramic (J) 300-mil DIPs**

SN54HC138 . . . J OR W PACKAGE
SN74HC138 . . . D, N, OR PW PACKAGE
(TOP VIEW)



SN54HC138 . . . FK PACKAGE
(TOP VIEW)



NC – No internal connection

description

The 'HC138 are designed to be used in high-performance memory-decoding or data-routing applications requiring very short propagation delay times. In high-performance memory systems, these decoders can be used to minimize the effects of system decoding. When employed with high-speed memories utilizing a fast enable circuit, the delay times of these decoders and the enable time of the memory are usually less than the typical access time of the memory. This means that the effective system delay introduced by the decoders is negligible.

The conditions at the binary-select inputs at the three enable inputs select one of eight output lines. Two active-low and one active-high enable inputs reduce the need for external gates or inverters when expanding. A 24-line decoder can be implemented without external inverters and a 32-line decoder requires only one inverter. An enable input can be used as a data input for demultiplexing applications.

The SN54HC138 is characterized for operation over the full military temperature range of –55°C to 125°C. The SN74HC138 is characterized for operation from –40°C to 85°C.

Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

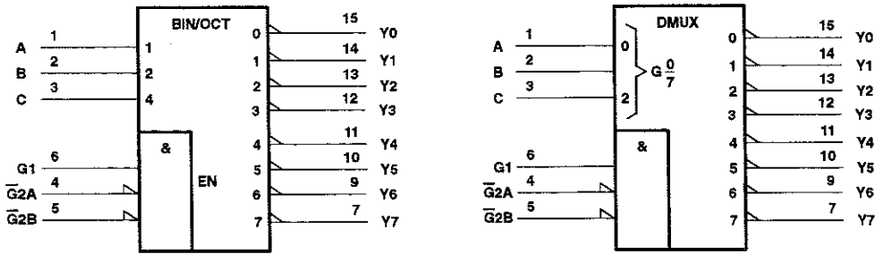
SN54HC138, SN74HC138
3-LINE TO 8-LINE DECODERS/DEMULTIPLEXERS

SCLS107C – DECEMBER 1982 – REVISED MAY 1997

FUNCTION TABLE

INPUTS						OUTPUTS							
ENABLE			SELECT										
G1	$\overline{G2A}$	$\overline{G2B}$	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	L	H	H	H	H	H	L	H	H	H	H
H	L	L	H	L	L	H	H	H	H	L	H	H	H
H	L	L	H	L	H	H	H	H	H	H	L	H	H
H	L	L	H	H	L	H	H	H	H	H	H	L	H
H	L	L	H	H	H	H	H	H	H	H	H	H	L

logic symbols (alternatives)†

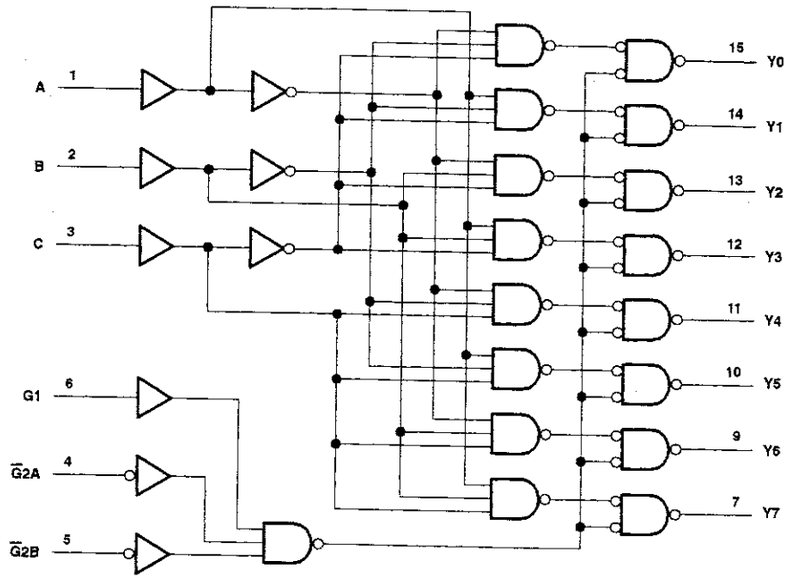


† These symbols are in accordance with ANSI/IEEE Std 91-1984 and IEC Publication 617-12.
Pin numbers shown are for the D, J, N, PW, and W packages.

SN54HC138, SN74HC138
3-LINE TO 8-LINE DECODERS/DEMULTIPLEXERS

SCLS107C – DECEMBER 1982 – REVISED MAY 1997

logic diagram (positive logic)



Pin numbers shown are for the D, J, N, PW, and W packages.

absolute maximum ratings over operating free-air temperature range†

Supply voltage range, V_{CC} ..... –0.5 V to 7 V
Input clamp current, I_{IK} ($V_I < 0$ or $V_I > V_{CC}$) (see Note 1) ..... ±20 mA
Output clamp current, I_{OK} ($V_O < 0$ or $V_O > V_{CC}$) (see Note 1) ..... ±20 mA
Continuous output current, I_O ($V_O = 0$ to V_{CC}) ..... ±25 mA
Continuous current through V_{CC} or GND ..... ±50 mA
Package thermal impedance, θ_{JA} (see Note 2): D package ..... 113°C/W
N package ..... 78°C/W
PW package ..... 149°C/W
Storage temperature range, T_{stg} ..... –65°C to 150°C

† Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTES: 1. The input and output voltage ratings may be exceeded if the input and output current ratings are observed.
2. The package thermal impedance is calculated in accordance with JESD 51, except for through-hole packages, which use a trace length of zero.

SN54HC138, SN74HC138
3-LINE TO 8-LINE DECODERS/DEMULTIPLEXERS

SCLS107C – DECEMBER 1982 – REVISED MAY 1997

recommended operating conditions

			SN54HC138			SN74HC138			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	
V_{CC}	Supply voltage		2	5	6	2	5	6	V
V_{IH}	High-level input voltage	V_{CC} = 2 V	1.5			1.5			V
		V_{CC} = 4.5 V	3.15			3.15			
		V_{CC} = 6 V	4.2			4.2			
V_{IL}	Low-level input voltage	V_{CC} = 2 V	0		0.5	0		0.5	V
		V_{CC} = 4.5 V	0		1.35	0		1.35	
		V_{CC} = 6 V	0		1.8	0		1.8	
V_I	Input voltage		0		V_{CC}	0		V_{CC}	V
V_O	Output voltage		0		V_{CC}	0		V_{CC}	V
t_t	Input transition (rise and fall) time	V_{CC} = 2 V	0		1000	0		1000	ns
		V_{CC} = 4.5 V	0		500	0		500	
		V_{CC} = 6 V	0		400	0		400	
T_A	Operating free-air temperature		–55		125	–40		85	°C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER	TEST CONDITIONS		V_{CC}	T_A = 25°C			SN54HC138		SN74HC138		UNIT
				MIN	TYP	MAX	MIN	MAX	MIN	MAX	
V_{OH}	V_I = V_{IH} or V_{IL}	I_{OH} = –20 μA	2 V	1.9	1.998		1.9		1.9		V
			4.5 V	4.4	4.499		4.4		4.4		
			6 V	5.9	5.999		5.9		5.9		
		I_{OH} = –4 mA	4.5 V	3.98	4.3		3.7		3.84		
		I_{OH} = –5.2 mA	6 V	5.48	5.8		5.2		5.34		
V_{OL}	V_I = V_{IH} or V_{IL}	I_{OL} = 20 μA	2 V		0.002	0.1	0.1		0.1		V
			4.5 V		0.001	0.1	0.1		0.1		
			6 V		0.001	0.1	0.1		0.1		
		I_{OL} = 4 mA	4.5 V		0.17	0.26	0.4		0.33		
		I_{OL} = 5.2 mA	6 V		0.15	0.26	0.4		0.33		
I_I	V_I = V_{CC} or 0		6 V		±0.1	±100	±1000		±1000		nA
I_{CC}	V_I = V_{CC} or 0,	I_O = 0	6 V			8	160		80		μA
C_i			2 V to 6 V		3	10	10		10		pF

SN54HC138, SN74HC138
3-LINE TO 8-LINE DECODERS/DEMULTIPLEXERS

SCLS107C – DECEMBER 1982 – REVISED MAY 1997

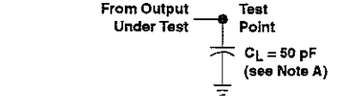
switching characteristics over recommended operating free-air temperature range, C_L = 50 pF (unless otherwise noted) (see Figure 1)

PARAMETER	FROM (INPUT)	TO (OUTPUT)	V_{CC}	T_A = 25°C			SN54HC138		SN74HC138		UNIT
				MIN	TYP	MAX	MIN	MAX	MIN	MAX	
t_{pd}	A, B, or C	Any Y	2 V		67	180		270		225	ns
			4.5 V		18	36		54		45	
			6 V		15	31		46		38	
	Enable	Any Y	2 V		66	155		235		195	
			4.5 V		18	31		47		39	
			6 V		15	26		40		33	
t_t		Any	2 V		38	75		110		95	ns
			4.5 V		8	15		22		19	
			6 V		6	13		19		16	

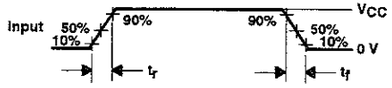
operating characteristics, T_A = 25°C

PARAMETER		TEST CONDITIONS	TYP	UNIT
C_{pd}	Power dissipation capacitance	No load	85	pF

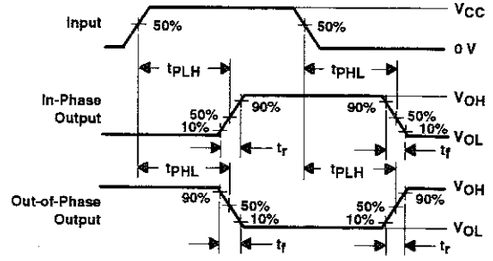
PARAMETER MEASUREMENT INFORMATION



LOAD CIRCUIT



VOLTAGE WAVEFORM
INPUT RISE AND FALL TIMES



VOLTAGE WAVEFORMS
PROPAGATION DELAY AND OUTPUT TRANSITION TIMES

NOTES: A. C_L includes probe and test-fixture capacitance.
B. Phase relationships between waveforms were chosen arbitrarily. All input pulses are supplied by generators having the following characteristics: PRR ≤ 1 MHz, Z_O = 50 Ω, t_r = 6 ns, t_f = 6 ns.
C. The outputs are measured one at a time with one input transition per measurement.
D. t_{PLH} and t_{PHL} are the same as t_{pd}.

Figure 1. Load Circuit and Voltage Waveforms

TEXAS INSTRUMENTS

Data sheet acquired from Harris Semiconductor
SCHS150

September 1997

CD74HC151, CD74HCT151

High Speed CMOS Logic 8-Input Multiplexer

Features

- Complementary Data Outputs
- Buffered Inputs and Outputs
- Fanout (Over Temperature Range)
  - Standard Outputs . . . . . 10 LSTTL Loads
  - Bus Driver Outputs . . . . . 15 LSTTL Loads
- Wide Operating Temperature Range . . . -55°C to 125°C
- Balanced Propagation Delay and Transition Times
- Significant Power Reduction Compared to LSTTL Logic ICs
- Alternate Source is Philips/Signetics
- HC Types
  - 2V to 6V Operation
  - High Noise Immunity: N_{IL} = 30%, N_{IH} = 30%of V_{CC} at V_{CC} = 5V
- HCT Types
  - 4.5V to 5.5V Operation
  - Direct LSTTL Input Logic Compatibility, V_{IL} = 0.8V (Max), V_{IH} = 2V (Min)
  - CMOS Input Compatibility, $I_I \leq 1\mu A$ at V_{OL}, V_{OH}

Description

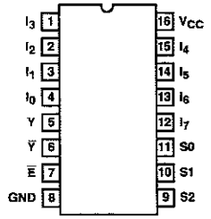
The Harris CD74HC151 and CD74HCT151 are single 8-channel digital multiplexers having three binary control inputs, S0, S1 and S2 and an active low enable (E) input. The three binary signals select 1 of 8 channels. Outputs are both inverting ($\overline{Y}$) and non-inverting (Y).

Ordering Information

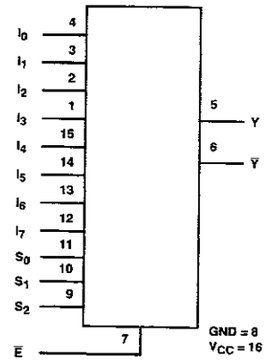
PART NUMBER	TEMP. RANGE (°C)	PACKAGE	PKG. NO.

Pinout

CD74HC151, CD74HCT151
(PDIP, SOIC)
TOP VIEW



Functional Diagram



TRUTH TABLE

SELECT INPUTS			DATA INPUTS								ENABLE	OUTPUT	
S2	S1	S0	I0	I1	I2	I3	I4	I5	I6	I7	E̅	Y̅	Y
X	X	X	X	X	X	X	X	X	X	X	H	H	L
L	L	L	L	X	X	X	X	X	X	X	L	H	L
L	L	L	H	X	X	X	X	X	X	X	L	L	H
L	L	H	X	L	X	X	X	X	X	X	L	H	L
L	L	H	X	H	X	X	X	X	X	X	L	L	H
L	H	L	X	X	L	X	X	X	X	X	L	H	L
L	H	L	X	X	H	X	X	X	X	X	L	L	H
L	H	H	X	X	X	L	X	X	X	X	L	H	L
L	H	H	X	X	X	H	X	X	X	X	L	L	H
H	L	L	X	X	X	X	L	X	X	X	L	H	L
H	L	L	X	X	X	X	H	X	X	X	L	L	H
H	L	H	X	X	X	X	X	L	X	X	L	H	L
H	L	H	X	X	X	X	X	H	X	X	L	L	H
H	H	L	X	X	X	X	X	X	L	X	L	H	L
H	H	L	X	X	X	X	X	X	H	X	L	L	H
H	H	H	X	X	X	X	X	X	X	L	L	H	L
H	H	H	X	X	X	X	X	X	X	H	L	L	H

NOTE: H = High Voltage Level, L = Low Voltage Level, X = Don't Care

CD74HC151, CD74HCT151

Absolute Maximum Ratings

DC Supply Voltage, V_{CC} ..... -0.5V to 7V
DC Input Diode Current, I_{IK}
For V_I < -0.5V or V_I > V_{CC} + 0.5V ..... ±20mA
DC Output Diode Current, I_{OK}
For V_O < -0.5V or V_O > V_{CC} + 0.5V ..... ±20mA
DC Output Source or Sink Current per Output Pin, I_O
For V_O > -0.5V or V_O < V_{CC} + 0.5V ..... ±25mA
DC V_{CC} or Ground Current, I_{CC} or I_{GND} ..... ±50mA

Operating Conditions

Temperature Range (T_A) ..... -55°C to 125°C
Supply Voltage Range, V_{CC}
HC Types ..... 2V to 6V
HCT Types ..... 4.5V to 5.5V
DC Input or Output Voltage, V_I, V_O ..... 0V to V_{CC}
Input Rise and Fall Time
2V ..... 1000ns (Max)
4.5V ..... 500ns (Max)
6V ..... 400ns (Max)

Thermal Information

Thermal Resistance (Typical, Note 3)	θ_{JA} (°C/W)
PDIP Package	90
SOIC Package	115

Maximum Junction Temperature ..... 150°C
Maximum Storage Temperature Range ..... -65°C to 150°C
Maximum Lead Temperature (Soldering 10s) ..... 300°C
(SOIC - Lead Tips Only)

CAUTION: Stresses above those listed in "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

NOTE:

3. θ_{JA} is measured with the component mounted on an evaluation PC board in free air.

DC Electrical Specifications

PARAMETER	SYMBOL	TEST CONDITIONS V_I (V)	TEST CONDITIONS I_O (mA)	V_{CC} (V)	25°C MIN	25°C TYP	25°C MAX	-40°C TO 85°C MIN	-40°C TO 85°C MAX	-55°C TO 125°C MIN	-55°C TO 125°C MAX	UNITS
HC TYPES												
High Level Input Voltage	V_{IH}	-	-	2	1.5	-	-	1.5	-	1.5	-	V
				4.5	3.15	-	-	3.15	-	3.15	-	V
				6	4.2	-	-	4.2	-	4.2	-	V
Low Level Input Voltage	V_{IL}	-	-	2	-	-	0.5	-	0.5	-	0.5	V
				4.5	-	-	1.35	-	1.35	-	1.35	V
				6	-	-	1.8	-	1.8	-	1.8	V
High Level Output Voltage CMOS Loads	V_{OH}	V_{IH} or V_{IL}	-0.02	2	1.9	-	-	1.9	-	1.9	-	V
			-0.02	4.5	4.4	-	-	4.4	-	4.4	-	V
			-0.02	6	5.9	-	-	5.9	-	5.9	-	V
High Level Output Voltage TTL Loads			-	-	-	-	-	-	-	-	-	V
			-4	4.5	3.98	-	-	3.84	-	3.7	-	V
			-5.2	6	5.48	-	-	5.34	-	5.2	-	V
Low Level Output Voltage CMOS Loads	V_{OL}	V_{IH} or V_{IL}	0.02	2	-	-	0.1	-	0.1	-	0.1	V
			0.02	4.5	-	-	0.1	-	0.1	-	0.1	V
			0.02	6	-	-	0.1	-	0.1	-	0.1	V
Low Level Output Voltage TTL Loads			-	-	-	-	-	-	-	-	-	V
			4	4.5	-	-	0.26	-	0.33	-	0.4	V
			5.2	6	-	-	0.26	-	0.33	-	0.4	V
Input Leakage Current	I_I	V_{CC} or GND	-	6	-	-	±0.1	-	±1	-	±1	µA
Quiescent Device Current	I_{CC}	V_{CC} or GND	0	6	-	-	8	-	80	-	160	µA

CD74HC151, CD74HCT151

DC Electrical Specifications (Continued)

PARAMETER	SYMBOL	TEST CONDITIONS V_I (V)	TEST CONDITIONS I_O (mA)	V_{CC} (V)	25°C MIN	25°C TYP	25°C MAX	-40°C TO 85°C MIN	-40°C TO 85°C MAX	-55°C TO 125°C MIN	-55°C TO 125°C MAX	UNITS
HCT TYPES												
High Level Input Voltage	V_{IH}	-	-	4.5 to 5.5	2	-	-	2	-	2	-	V
Low Level Input Voltage	V_{IL}	-	-	4.5 to 5.5	-	-	0.8	-	0.8	-	0.8	V
High Level Output Voltage CMOS Loads	V_{OH}	V_{IH} or V_{IL}	-0.02	4.5	4.4	-	-	4.4	-	4.4	-	V
High Level Output Voltage TTL Loads			-4	4.5	3.98	-	-	3.84	-	3.7	-	V
Low Level Output Voltage CMOS Loads	V_{OL}	V_{IH} or V_{IL}	0.02	4.5	-	-	0.1	-	0.1	-	0.1	V
Low Level Output Voltage TTL Loads			4	4.5	-	-	0.26	-	0.33	-	0.4	V
Input Leakage Current	I_I	V_{CC} and GND	0	5.5	-		±0.1	-	±1	-	±1	µA
Quiescent Device Current	I_{CC}	V_{CC} or GND	0	5.5	-	-	8	-	80	-	160	µA
Additional Quiescent Device Current Per Input Pin: 1 Unit Load	ΔI_{CC}	V_{CC} -2.1	-	4.5 to 5.5	-	100	360	-	450	-	490	µA

NOTE: For dual-supply systems theoretical worst case (V_I = 2.4V, V_{CC} = 5.5V) specification is 1.8mA.

HCT Input Loading Table

INPUT	UNIT LOADS
Select	1.5
Data	0.45
Enable	0.3

NOTE: Unit Load is ΔI_{CC} limit specified in DC Electrical Table, e.g., 360µA max at 25°C.

Switching Specifications Input t_r, t_f = 6ns

PARAMETER	SYMBOL	TEST CONDITIONS	V_{CC} (V)	25°C MIN	25°C TYP	25°C MAX	-40°C TO 85°C MIN	-40°C TO 85°C MAX	-55°C TO 125°C MIN	-55°C TO 125°C MAX	UNITS
HC TYPES											
Propagation Delay (Figure 1) Any Data Input to Y	t_{PLH}, t_{PHL}	C_L = 50pF	2	-	-	170	-	215	-	255	ns
			4.5	-	-	34	-	43	-	51	ns
		C_L =15pF	5	-	14	-	-	-	-	-	ns
		C_L = 50pF	6	-	-	29	-	37	-	43	ns

CD74HC151, CD74HCT151

Switching Specifications Input t_r, t_f = 6ns **(Continued)**

PARAMETER	SYMBOL	TEST CONDITIONS	V_{CC} (V)	25°C MIN	25°C TYP	25°C MAX	-40°C TO 85°C MIN	-40°C TO 85°C MAX	-55°C TO 125°C MIN	-55°C TO 125°C MAX	UNITS
Any Data Input to $\overline{Y}$	t_{PLH}, t_{PHL}	C_L = 50pF	2	-	-	185	-	230	-	280	ns
			4.5	-	-	37	-	46	-	56	ns
		C_L =15pF	5	-	15	-	-	-	-	-	ns
		C_L = 50pF	6	-	-	31	-	39	-	48	ns
Any Select to Y	t_{PLH}, t_{PHL}	C_L = 50pF	2	-	-	185	-	230	-	280	ns
			4.5	-	-	37	-	46	-	56	ns
		C_L =15pF	5	-	15	-	-	-	-	-	ns
		C_L = 50pF	6	-	-	31	-	39	-	48	ns
Any Select to $\overline{Y}$	t_{PLH}, t_{PHL}	C_L = 50pF	2	-	-	205	-	255	-	310	ns
			4.5	-	-	41	-	51	-	62	ns
		C_L =15pF	5	-	17	-	-	-	-	-	ns
		C_L = 50pF	6	-	-	35	-	43	-	53	ns
Enable to Y	t_{PLH}, t_{PHL}	C_L = 50pF	2	-	-	140	-	175	-	210	ns
			4.5	-	-	28	-	35	-	42	ns
		C_L =15pF	5	-	11	-	-	-	-	-	ns
		C_L = 50pF	6	-	-	24	-	30	-	36	ns
Enable to $\overline{Y}$	t_{PLH}, t_{PHL}	C_L = 50pF	2	-	-	145	-	180	-	220	ns
			4.5	-	-	29	-	36	-	44	ns
		C_L =15pF	5	-	12	-	-	-	-	-	ns
		C_L = 50pF	6	-	-	25	-	31	-	38	ns
Output Transition Time (Figure 1)	t_{TLH}, t_{THL}	C_L = 50pF	2	-	-	75	-	95	-	110	ns
			4.5	-	-	15	-	19	-	22	ns
			6	-	-	13	-	16	-	19	ns
Input Capacitance	C_{IN}	-	-	-	-	10	-	10	-	10	pF
Power Dissipation Capacitance (Notes 4, 5)	C_{PD}	-	5	-	59	-	-	-	-	-	pF
HCT TYPES											
Propagation Delay (Figure 2) Any Data Input to Y	t_{PLH}, t_{PHL}	C_L = 50pF	4.5	-	-	38	-	48	-	57	ns
		C_L =15pF	5	-	16	-	-		-	-	ns
Any Data Input to $\overline{Y}$	t_{PLH}, t_{PHL}	C_L = 50pF	4.5	-	-	36	-	45	-	54	ns
		C_L =15pF	5	-	15	-	-	-	-	-	ns
Any Select to Y	t_{PLH}, t_{PHL}	C_L = 50pF	4.5	-		41	-	51	-	62	ns
		C_L =15pF	5	-	17	-	-	-	-	-	ns
Any Select to $\overline{Y}$	t_{PLH}, t_{PHL}	C_L = 50pF	4.5	-	-	43	-	54	-	65	ns
		C_L =15pF	5	-	18	-	-	-	-	-	ns
Enable to Y	t_{PLH}, t_{PHL}	C_L = 50pF	4.5	-	-	29	-	36	-	44	ns
		C_L =15pF	5	-	12	-	-	-	-	-	ns

CD74HC151, CD74HCT151

Switching Specifications Input t_r, t_f = 6ns **(Continued)**

PARAMETER	SYMBOL	TEST CONDITIONS	V_{CC} (V)	25°C MIN	25°C TYP	25°C MAX	-40°C TO 85°C MIN	-40°C TO 85°C MAX	-55°C TO 125°C MIN	-55°C TO 125°C MAX	UNITS
Enable to $\overline{Y}$	C_L = 50pF	C_L = 50pF	4.5	-	-	36	-	46	-	54	ns
	C_L =15pF	C_L =15pF	5	15	-	-	-	-	-	-	ns
Output Transition Time	t_{TLH}, t_{THL}	C_L = 50pF	4.5	-	-	15	-	19	-	22	ns
Input Capacitance	C_{IN}	-	-	-	-	10	-	10	-	10	pF
Power Dissipation Capacitance (Notes 4, 5)	C_{PD}	-	5		58	-	-	-	-	-	pF

NOTES:

4. C_{PD} is used to determine the dynamic power consumption, per gate.
5. $P_D = V_{CC}^2 f_i (C_{PD} + C_L)$ where f_i = input frequency, C_L = output load capacitance, V_{CC} = supply voltage.

Test Circuit and Waveform

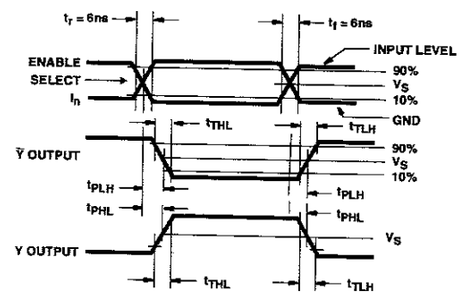


FIGURE 1.

TEXAS INSTRUMENTS

Data sheet acquired from Harris Semiconductor
SCHS154

February 1998

CD74HC161, CD74HCT161, CD74HC163, CD74HCT163

High Speed CMOS Logic Presettable Counters

Features

- **CD74HC161, CD74HCT161 4-Bit Binary Counter, Asynchronous Reset**
- **CD74HC163, CD74HCT163 4-Bit Binary Counter, Synchronous Reset**
- **Synchronous Counting and Loading**
- **Two Count Enable Inputs for n-Bit Cascading**
- **Look-Ahead Carry for High-Speed Counting**
- **Fanout (Over Temperature Range)**
  - **Standard Outputs . . . . . 10 LSTTL Loads**
  - **Bus Driver Outputs . . . . . 15 LSTTL Loads**
- **Wide Operating Temperature Range . . . -55°C to 125°C**
- **Balanced Propagation Delay and Transition Times**
- **Significant Power Reduction Compared to LSTTL Logic ICs**
- **HC Types**
  - **2V to 6V Operation**
  - **High Noise Immunity: N_{IL} = 30%, N_{IH} = 30% of V_{CC} at V_{CC} = 5V**
- **HCT Types**
  - **4.5V to 5.5V Operation**
  - **Direct LSTTL Input Logic Compatibility, V_{IL} = 0.8V (Max), V_{IH} = 2V (Min)**
  - **CMOS Input Compatibility, $I_I \leq 1\mu A$ at V_{OL}, V_{OH}**

Ordering Information

PART NUMBER	TEMP. RANGE (°C)	PACKAGE	PKG. NO.
CD74HC161E	-55 to 125	16 Ld PDIP	E16.3
CD74HC161M	-55 to 125	16 Ld SOIC	M16.15
CD74HC163E	-55 to 125	16 Ld PDIP	E16.3
CD74HC163M	-55 to 125	16 Ld SOIC	M16.15
CD74HCT161E	-55 to 125	16 Ld PDIP	E16.3
CD74HCT161M	-55 to 125	16 Ld SOIC	M16.15
CD74HCT163E	-55 to 125	16 Ld PDIP	E16.3
CD74HCT163M	-55 to 125	16 Ld SOIC	M16.15

NOTES:

1. When ordering, use the entire part number. Add the suffix 96 to obtain the variant in the tape and reel.
2. Wafer and die for this part number is available which meets all electrical specifications. Please contact your local sales office or Harris customer service for ordering information.

Description

The Harris CD74HC161, CD74HCT161, CD74HC163 and CD74HCT163 are presettable synchronous counters that feature look-ahead carry logic for use in high-speed counting applications. The CD74HC161 and CD74HCT161 are asynchronous reset decade and binary counters, respectively; the CD74HC163 and CD74HCT163 devices decade and binary counters, respectively and are reset synchronously with the clock. Counting and parallel presetting are both accomplished synchronously with the negative-to-positive transition of the clock.

A low level on the synchronous parallel enable input, $\overline{SPE}$, disables counting operation and allows data at the P0 to P3 inputs to be loaded into the counter (provided that the setup and hold requirements for $\overline{SPE}$ are met).

All counters are reset with a low level on the Master Reset input, MR. In the CD74HC163 and CD74HCT163 counters (synchronous reset types), the requirements for setup and hold time with respect to the clock must be met.

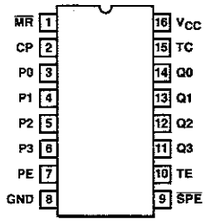
Two count enables, PE and TE, in each counter are provided for n-bit cascading. In all counters reset action occurs regardless of the level of the $\overline{SPE}$, PE and TE inputs (and the clock input, CP, in the CD74HC161 and CD74HCT161 types).

If a decade counter is preset to an illegal state or assumes an illegal state when power is applied, it will return to the normal sequence in one count as shown in state diagram.

The look-ahead carry feature simplifies serial cascading of the counters. Both count enable inputs (PE and TE) must be high to count. The TE input is gated with the Q outputs of all four stages so that at the maximum count the terminal count (TC) output goes high for one clock period. This TC pulse is used to enable the next cascaded stage.

Pinout

CD74HC161, CD74HCT161, CD74HC163, CD74HCT163
(PDIP, SOIC)
TOP VIEW



Absolute Maximum Ratings

DC Supply Voltage, V_{CC} . . . . . -0.5V to 7V
DC Input Diode Current, I_{IK}
For V_I < -0.5V or V_I > V_{CC} + 0.5V . . . . . ±20mA
DC Output Diode Current, I_{OK}
For V_O < -0.5V or V_O > V_{CC} + 0.5V . . . . . ±20mA
DC Drain Current, per Output, I_O
For -0.5V < V_O < V_{CC} + 0.5V . . . . . ±25mA
DC Output Source or Sink Current per Output Pin, I_O
For V_O > -0.5V or V_O < V_{CC} + 0.5V . . . . . ±25mA
DC V_{CC} or Ground Current, I_{CC} . . . . . ±50mA

Operating Conditions

Temperature Range, T_A . . . . . -55°C to 125°C
Supply Voltage Range, V_{CC}
HC Types . . . . . 2V to 6V
HCT Types . . . . . 4.5V to 5.5V
DC Input or Output Voltage, V_I, V_O . . . . . 0V to V_{CC}
Input Rise and Fall Time
2V . . . . . 1000ns (Max)
4.5V . . . . . 500ns (Max)
6V . . . . . 400ns (Max)

Thermal Information

Thermal Resistance (Typical, Note 6)	θ_{JA} (°C/W)
PDIP Package	90
SOIC Package	160

Maximum Junction Temperature . . . . . 150°C
Maximum Storage Temperature Range . . . . . -65°C to 150°C
Maximum Lead Temperature (Soldering 10s) . . . . . 300°C
(SOIC - Lead Tips Only)

CAUTION: Stresses above those listed in "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

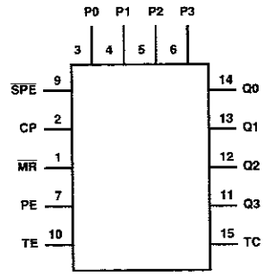
NOTE:

6. θ_{JA} is measured with the component mounted on an evaluation PC board in free air.

DC Electrical Specifications

PARAMETER	SYMBOL	TEST CONDITIONS V_I (V)	TEST CONDITIONS I_O (mA)	V_{CC} (V)	25°C MIN	25°C TYP	25°C MAX	-40°C TO 85°C MIN	-40°C TO 85°C MAX	-55°C TO 125°C MIN	-55°C TO 125°C MAX	UNITS
HC TYPES												
High Level Input Voltage	V_{IH}	-	-	2	1.5	-	-	1.5	-	1.5	-	V
				4.5	3.15	-	-	3.15	-	3.15	-	V
				6	4.2	-	-	4.2	-	4.2	-	V
Low Level Input Voltage	V_{IL}	-	-	2	-	-	0.5	-	0.5	-	0.5	V
				4.5	-	-	1.35	-	1.35	-	1.35	V
				6	-	-	1.8	-	1.8	-	1.8	V
High Level Output Voltage CMOS Loads	V_{OH}	V_{IH} or V_{IL}	-0.02	2	1.9	-	-	1.9	-	1.9	-	V
			-0.02	4.5	4.4	-	-	4.4	-	4.4	-	V
			-0.02	6	5.9	-	-	5.9	-	5.9	-	V
High Level Output Voltage TTL Loads			-	-	-	-	-	-	-	-	-	V
			-4	4.5	3.98	-	-	3.84	-	3.7	-	V
			-5.2	6	5.48	-	-	5.34	-	5.2	-	V
Low Level Output Voltage CMOS Loads	V_{OL}	V_{IH} or V_{IL}	0.02	2	-	-	0.1	-	0.1	-	0.1	V
			0.02	4.5	-	-	0.1	-	0.1	-	0.1	V
			0.02	6	-	-	0.1	-	0.1	-	0.1	V
Low Level Output Voltage TTL Loads			-	-	-	-	-	-	-	-	-	V
			4	4.5	-	-	0.26	-	0.33	-	0.4	V
			5.2	6	-	-	0.26	-	0.33	-	0.4	V
Input Leakage Current	I_I	V_{CC} or GND	-	6	-	-	±0.1	-	±1	-	±1	µA

Functional Diagram



MODE SELECT - FUNCTION TABLE FOR CD74HC/HCT161

OPERATING MODE	INPUTS						OUTPUTS	
	$\overline{MR}$	CP	PE	TE	$\overline{SPE}$	P_n	Q_n	TC
Reset (Clear)	L	X	X	X	X	X	L	L
Parallel Load	H	↑	X	X	l	l	L	L
	H	↑	X	X	l	h	H	(Note 3)
Count	H	↑	h	h	h (Note 5)	X	Count	(Note 3)
Inhibit	H	X	l (Note 4)	X	h (Note 5)	X	q_n	(Note 3)
	H	X	X	l (Note 4)	h (Note 5)	X	q_n	L

MODE SELECT - FUNCTION TABLE FOR CD74HC/HCT163

OPERATING MODE	INPUTS						OUTPUTS	
	$\overline{MR}$	CP	PE	TE	$\overline{SPE}$	P_n	Q_n	TC
Reset (Clear)	l	↑	X	X	X	X	L	L
Parallel Load	h (Note 5)	↑	X	X	l	l	L	L
	h (Note 5)	↑	X	X	l	h	H	(Note 3)
Count	h (Note 5)	↑	h	h	h (Note 5)	X	Count	(Note 3)
Inhibit	h (Note 5)	X	l (Note 4)	X	h (Note 5)	X	q_n	(Note 3)
	h (Note 5)	X	X	l (Note 4)	h (Note 5)	X	q_n	L

NOTE: H = High voltage level steady state; L = Low voltage level steady state; h = High voltage level one setup time prior to the Low-to-High clock transition; l = Low voltage level one setup time prior to the Low-to-High clock transition; X = Don't Care; q = Lower case letters indicate the state of the referenced output prior to the Low-to-High clock transition; ↑ = Low-to-High clock transition.

3. The TC output is High when TE is High and the counter is at Terminal Count (HHHH for CD74HC/HCT161 and CD74HC/HCT163).
4. The High-to-Low transition of PE or TE on the CD74HC/HCT161 and the CD74HC/HCT163 should only occur while CP is HIGH for conventional operation.
5. The Low-to-High transition of $\overline{SPE}$ on the CD74HC/HCT161 and $\overline{SPE}$ or $\overline{MR}$ on the CD74HC/HCT163 should only occur while CP is HIGH for conventional operation.

DC Electrical Specifications (Continued)

PARAMETER	SYMBOL	TEST CONDITIONS V_I (V)	TEST CONDITIONS I_O (mA)	V_{CC} (V)	25°C MIN	25°C TYP	25°C MAX	-40°C TO 85°C MIN	-40°C TO 85°C MAX	-55°C TO 125°C MIN	-55°C TO 125°C MAX	UNITS
Quiescent Device Current	I_{CC}	V_{CC} or GND	0	6	-	-	8	-	80	-	160	µA
HCT TYPES												
High Level Input Voltage	V_{IH}	-	-	4.5 to 5.5	2	-	-	2	-	2	-	V
Low Level Input Voltage	V_{IL}	-	-	4.5 to 5.5	-	-	0.8	-	0.8	-	0.8	V
High Level Output Voltage CMOS Loads	V_{OH}	V_{IH} or V_{IL}	-0.02	4.5	4.4	-	-	4.4	-	4.4	-	V
High Level Output Voltage TTL Loads			-4	4.5	3.98	-	-	3.84	-	3.7	-	V
Low Level Output Voltage CMOS Loads	V_{OL}	V_{IH} or V_{IL}	0.02	4.5	-	-	0.1	-	0.1	-	0.1	V
Low Level Output Voltage TTL Loads			4	4.5	-	-	0.26	-	0.33	-	0.4	V
Input Leakage Current	I_I	V_{CC} and GND	0	5.5	-		±0.1	-	±1	-	±1	µA
Quiescent Device Current	I_{CC}	V_{CC} or GND	0	5.5	-	-	8	-	80	-	160	µA
Additional Quiescent Device Current Per Input Pin: 1 Unit Load	ΔI_{CC} (Note)	V_{CC} -2.1	-	4.5 to 5.5	-	100	360	-	450	-	490	µA

NOTE: For dual-supply systems theoretical worst case (V_I = 2.4V, V_{CC} = 5.5V) specification is 1.8mA.

HCT Input Loading Table

INPUT	UNIT LOADS
P0 - P3	0.25
PE	0.65
CP	1.05
$\overline{MR}$	0.8
$\overline{SPE}$	0.5
TE	1.05

NOTE: Unit Load is ΔI_{CC} limit specified in DC Electrical Table, e.g., 360µA max at 25°C.

Prerequisite For Switching Specifications

PARAMETER	SYMBOL	TEST CONDITIONS	V_{CC} (V)	25°C MIN	25°C TYP	25°C MAX	-40°C TO 85°C MIN	-40°C TO 85°C MAX	-55°C TO 125°C MIN	-55°C TO 125°C MAX	UNITS
HC TYPES											
Maximum CP Frequency (Note7)	f_{MAX}	-	2	6	-	-	5	-	4	-	MHz
			4.5	30	-	-	24	-	20	-	MHz
			6	35	-	-	28	-	24	-	MHz

Prerequisite For Switching Specifications (Continued)

PARAMETER	SYMBOL	TEST CONDITIONS	V_{CC} (V)	25°C MIN	25°C TYP	25°C MAX	-40°C TO 85°C MIN	-40°C TO 85°C MAX	-55°C TO 125°C MIN	-55°C TO 125°C MAX	UNITS
CP Width (Low)	$t_{W(L)}$	-	2	80	-	-	100	-	120	-	ns
			4.5	16	-	-	20	-	24	-	ns
			6	14	-	-	17	-	20	-	ns
$\overline{MR}$ Pulse Width (161)	t_W	-	2	100	-	-	125	-	150	-	ns
			4.5	20	-	-	25	-	30	-	ns
			6	17	-	-	21	-	26	-	ns
Setup Time, Pn to CP	t_{SU}	-	2	60	-	-	75	-	90	-	ns
			4.5	12	-	-	15	-	18	-	ns
			6	10	-	-	13	-	15	-	ns
Setup Time, PE or TE to CP	t_{SU}	-	2	50	-	-	65	-	75	-	ns
			4.5	10	-	-	13	-	15	-	ns
			6	9	-	-	11	-	13	-	ns
Setup Time, $\overline{SPE}$ to CP	t_{SU}	-	2	60	-	-	75	-	90	-	ns
			4.5	12	-	-	15	-	18	-	ns
			6	10	-	-	13	-	15	-	ns
Setup Time, $\overline{MR}$ to CP (163)	t_{SU}	-	2	65	-	-	80	-	100	-	ns
			4.5	13	-	-	16	-	20	-	ns
			6	11	-	-	14	-	17	-	ns
Hold Time, PN to CP	t_H	-	2	3	-	-	3	-	3	-	ns
			4.5	3	-	-	3	-	3	-	ns
			6	3	-	-	3	-	3	-	ns
Hold Time, TE or PE to CP	t_H	-	2	0	-	-	0	-	0	-	ns
			4.5	0	-	-	0	-	0	-	ns
			6	0	-	-	0	-	0	-	ns
Hold Time, $\overline{SPE}$ to CP	t_H	-	2	0	-	-	0	-	0	-	ns
			4.5	0	-	-	0	-	0	-	ns
			6	0	-	-	0	-	0	-	ns
Recovery Time, $\overline{MR}$ to CP (161)	t_{REC}	-	2	75	-	-	95	-	110	-	ns
			4.5	15	-	-	19	-	22	-	ns
			6	13	-	-	16	-	19	-	ns
HCT TYPES											
Maximum CP Frequency	f_{MAX}	-	4.5	30	-	-	24	-	20	-	MHz
CP Width (Low) (Note 7)	$t_{W(L)}$	-	4.5	16	-	-	20	-	24	-	ns
$\overline{MR}$ Pulse Width (161)	t_W	-	4.5	20	-	-	25	-	30	-	ns
Setup Time, Pn to CP	t_{SU}	-	4.5	10	-	-	13	-	15	-	ns
Setup Time, PE or TE to CP	t_{SU}	-	4.5	13	-	-	16	-	20	-	ns
Setup Time, $\overline{SPE}$ to CP	t_{SU}	-	4.5	12	-	-	15	-	18	-	ns
Setup Time, $\overline{MR}$ to CP (163)	t_{SU}	-	4.5	13	-	-	16	-	20	-	ns
Hold Time, PN to CP	t_H	-	4.5	5	-	-	5	-	5	-	ns
Hold Time, TE or PE to CP	t_H	-	4.5	3	-	-	3	-	3	-	ns

Prerequisite For Switching Specifications (Continued)

PARAMETER	SYMBOL	TEST CONDITIONS	V_{CC} (V)	25°C MIN	25°C TYP	25°C MAX	-40°C TO 85°C MIN	-40°C TO 85°C MAX	-55°C TO 125°C MIN	-55°C TO 125°C MAX	UNITS
Hold Time, $\overline{SPE}$ to CP	t_H	-	4.5	3	-	-	3	-	3	-	ns
Recovery Time, $\overline{MR}$ to CP (161)	t_{REC}	-	4.5	15	-	-	19	-	22	-	ns

NOTE:

7. Applies to non-cascaded operation only. With cascaded counters clock to terminal count propagation delays, count enables (PE or TE)-to-clock setup times, and count enables (PE or TE)-to-clock hold times determine maximum clock frequency. For example with these HC devices:

$$f_{MAX}\,(CP) = \frac{1}{\text{CP-to-TC prop. delay} + \text{TE-to-CP setup} + \text{TE-to-CP Hold}} = \frac{1}{37 + 10 + 0} = 21\text{MHz(min)}$$

Switching Specifications $C_L = 50pF$, Input t_r, $t_f = 6ns$

PARAMETER	SYMBOL	TEST CONDITIONS	V_{CC} (V)	25°C MIN	25°C TYP	25°C MAX	-40°C TO 85°C MIN	-40°C TO 85°C MAX	-55°C TO 125°C MIN	-55°C TO 125°C MAX	UNITS
HC TYPES											
Propagation Delay CP to TC	t_{PHL}, t_{PLH}	$C_L = 50pF$	2	-	-	185	-	230	-	280	ns
			4.5	-	-	37	-	46	-	56	ns
		$C_L = 15pF$	5	-	15	-	-	-	-	-	ns
		$C_L = 50pF$	6	-	-	31	-	39	-	48	ns
CP to Qn	t_{PHL}, t_{PLH}	$C_L = 50pF$	2	-	-	185	-	230	-	280	ns
			4.5	-	-	37	-	46	-	56	ns
		$C_L = 15pF$	5	-	15	-	-	-	-	-	ns
		$C_L = 50pF$	6	-	-	31	-	39	-	48	ns
TE to TC	t_{PHL}, t_{PLH}	$C_L = 50pF$	2	-	-	120	-	150	-	180	ns
			4.5	-	-	24	-	30	-	36	ns
		$C_L = 15pF$	5	-	9	-	-	-	-	-	ns
		$C_L = 50pF$	6	-	-	20	-	26	-	31	ns
$\overline{MR}$ to Qn (161)	t_{PHL}	$C_L = 50pF$	2	-	-	210	-	265	-	315	ns
			4.5	-	-	42	-	53	-	63	ns
		$C_L = 15pF$	5	-	18	-	-	-	-	-	ns
		$C_L = 50pF$	6	-	-	36	-	45	-	54	ns
$\overline{MR}$ to TC (161)	t_{PHL}	$C_L = 50pF$	2	-	-	210	-	265	-	315	ns
			4.5	-	-	42	-	53	-	63	ns
		$C_L = 50pF$	6	-	-	36	-	45	-	54	ns
Output Transition Time	t_{THL}, t_{TLH}	$C_L = 50pF$	2	-	-	75	-	95	-	110	ns
			4.5	-	-	15	-	19	-	22	ns
			6	-	-	13	-	16	-	19	ns
Power Dissipation Capacitance (Notes 8, 9)	C_{PD}	-	5	-	60	-	-	-	-	-	pF

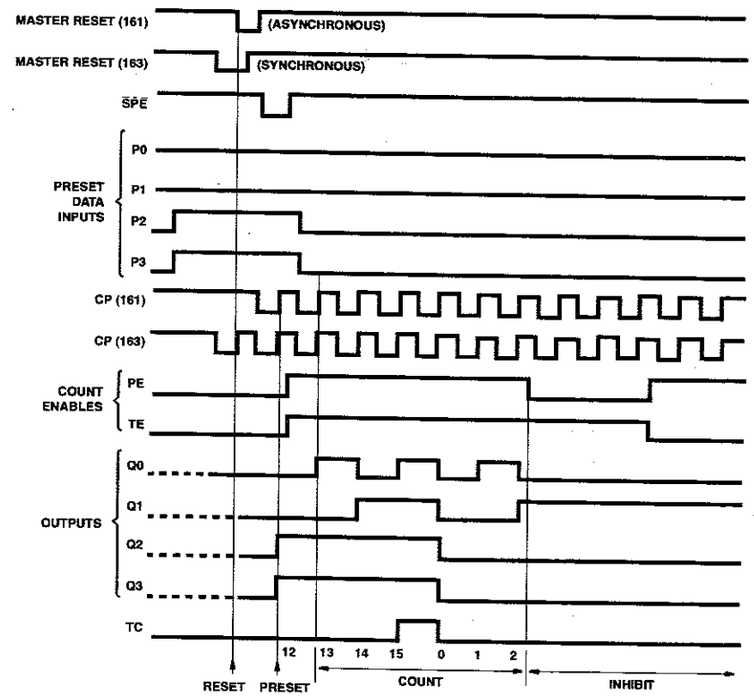
Switching Specifications $C_L = 50pF$, Input t_r, $t_f = 6ns$ **(Continued)**

PARAMETER	SYMBOL	TEST CONDITIONS	V_{CC} (V)	25°C MIN	25°C TYP	25°C MAX	-40°C TO 85°C MIN	-40°C TO 85°C MAX	-55°C TO 125°C MIN	-55°C TO 125°C MAX	UNITS
Input Capacitance	C_{IN}	$C_L = 50pF$	-	10	-	10	-	10	-	10	pF
HCT TYPES											
Propagation Delay CP to TC	t_{PHL}, t_{PLH}	$C_L = 50pF$	4.5	-	-	42	-	53	-	63	ns
		$C_L = 15pF$	5	-	18	-	-	-	-	-	ns
CP to Qn	t_{PHL}, t_{PLH}	$C_L = 50pF$	4.5	-	-	39	-	49	-	59	ns
		$C_L = 15pF$	5	-	16	-	-	-	-	-	ns
TE to TC	t_{PHL}, t_{PLH}	$C_L = 50pF$	4.5	-	-	32	-	40	-	48	ns
		$C_L = 15pF$	5	-	13	-	-	-	-	-	ns
$\overline{MR}$ to Qn (161)	t_{PHL}	$C_L = 50pF$	4.5	-	-	50	-	63	-	75	ns
		$C_L = 15pF$	5	-	21	-	-	-	-	-	ns
$\overline{MR}$ to TC (161)	t_{PHL}	$C_L = 50pF$	4.5	-	-	50	-	63	-	75	ns
Output Transition Time	t_{THL}, t_{TLH}	$C_L = 50pF$	4.5	-	-	15	-	19	-	22	ns
Power Dissipation Capacitance (Notes 6, 7)	C_{PD}	-	5	-	63	-	-	-	-	-	pF
Input Capacitance	C_{IN}	$C_L = 50pF$	-	10	-	10	-	10	-	10	pF

NOTES:

8. C_{PD} is used to determine the dynamic power consumption, per package.
9. $P_D = C_{PD} V_{CC}^2 f_i + \Sigma(C_L V_{CC}^2 f_O)$ where f_i = Input Frequency, f_O = Output Frequency, C_L = Output Load Capacitance, V_{CC} = Supply Voltage.

Timing Diagram



Sequence illustrated on waveforms:

1. Reset outputs to zero.
2. Preset to binary twelve.
3. Count to thirteen, fourteen, fifteen, zero, one, and two.
4. Inhibit.

TEXAS INSTRUMENTS

Data sheet acquired from Harris Semiconductor
SCHS156

February 1998

CD74HC165, CD74HCT165

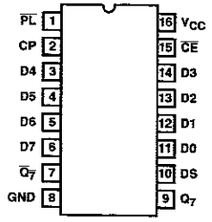
High Speed CMOS Logic 8-Bit Parallel-In/Serial-Out Shift Register

Features

- Buffered Inputs
- Asynchronous Parallel Load
- Complementary Outputs
- Typical f_{MAX} = 60MHz at V_{CC} = 5V, C_L = 15pF, T_A = 25°C
- Fanout (Over Temperature Range)
  - Standard Outputs . . . . . 10 LSTTL Loads
  - Bus Driver Outputs . . . . . 15 LSTTL Loads
- Wide Operating Temperature Range . . . -55°C to 125°C
- Balanced Propagation Delay and Transition Times
- Significant Power Reduction Compared to LSTTL Logic ICs
- HC Types
  - 2V to 6V Operation
  - High Noise Immunity: N_{IL} = 30%, N_{IH} = 30% of V_{CC} at V_{CC} = 5V
- HCT Types
  - 4.5V to 5.5V Operation
  - Direct LSTTL Input Logic Compatibility, V_{IL} = 0.8V (Max), V_{IH} = 2V (Min)
  - CMOS Input Compatibility, $I_I \leq 1\mu A$ at V_{OL}, V_{OH}

Pinout

CD74HC165, CD74HCT165
(PDIP, SOIC)
TOP VIEW



Description

The Harris CD74HC165 and CD74HCT165 are 8-bit parallel or serial-in shift registers with complementary serial outputs (Q_7 and $\overline{Q_7}$) available from the last stage. When the parallel load ($\overline{PL}$) input is LOW, parallel data from the D0 to D7 inputs are loaded into the register asynchronously. When the $\overline{PL}$ is HIGH, data enters the register serially at the DS input and shifts one place to the right ($Q_0 \rightarrow Q_1 \rightarrow Q_2$, etc.) with each positive-going clock transition. This feature allow parallel-to-serial converter expansion by typing the Q_7 output to the DS input of the succeeding device.

For predictable operation the LOW-to-HIGH transition of $\overline{CE}$ should only take place while CP is HIGH. Also, CP an d $\overline{CE}$ should be LOW before the LOW-to-HIGH transition of PL to prevent shifting the data when $\overline{PL}$ goes HIGH.

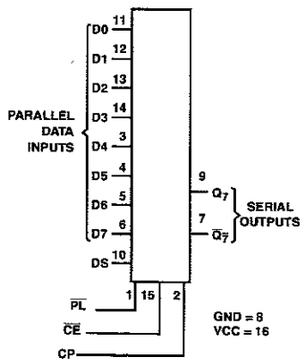
Ordering Information

PART NUMBER	TEMP. RANGE (°C)	PACKAGE	PKG. NO.
CD74HC165E	-55 to 125	16 Ld PDIP	E16.3
CD74HCT165E	-55 to 125	16 Ld PDIP	E16.3
CD74HC165M	-55 to 125	16 Ld SOIC	M16.15
CD74HCT165M	-55 to 125	16 Ld SOIC	M16.15

NOTES:

1. When ordering, use the entire part number. Add the suffix 96 to obtain the variant in the tape and reel.
2. Wafer and die is available which meets all electrical specifications. Please contact your local sales office or Harris customer service for ordering information.

Functional Diagram



TRUTH TABLE

OPERATING MODE	INPUTS					Q_n REGISTER		OUTPUTS	
	$\overline{PL}$	$\overline{CE}$	CP	DS	D0 - D7	Q_0	Q_1 - Q_6	Q_7	$\overline{Q_7}$
Parallel Load	L	X	X	X	L	L	L-L	L	H
	L	X	X	X	H	H	H-H	H	L
Serial Shift	H	L	↑	l	X	L	q_0 - q_5	q_6	$\overline{q_6}$
	H	L	↑	h	X	H	q_0 - q_5	q_6	$\overline{q_6}$
Hold Do Nothing	H	H	X	X	X	q_0	q_1 - q_6	q_7	$\overline{q_7}$

TRUTH TABLE

OPERATING MODE	INPUTS					Q_n REGISTER		OUTPUTS	
	$\overline{PL}$	$\overline{CE}$	CP	DS	D0 - D7	Q_0	Q_1 - Q_5	Q_7	$\overline{Q}_7$

NOTE:
H = High Voltage Level
h = High Voltage Level One Set-up Time Prior To The Low-to-high Clock Transition
l = Low Voltage Level One Set-up Time Prior To The Low-to-high Clock Transition
L = Low Voltage Level
X = Don't Care
↑ = Transition from Low to High Level
q_n = Lower Case Letters Indicate The State Of the Reference Output Clock Transition

Absolute Maximum Ratings

DC Supply Voltage, V_{CC} ..... -0.5V to 7V
DC Input Diode Current, I_{IK}
For V_I < -0.5V or V_I > V_{CC} + 0.5V ..... ±20mA
DC Output Diode Current, I_{OK}
For V_O < -0.5V or V_O > V_{CC} + 0.5V ..... ±20mA
DC Drain Current per Output, IO
For V_O < -0.5V V_O > V_{CC} + 0.5V ..... ±25mA
DC Output Source or Sink Current per Output Pin, I_O
For V_O > -0.5V or V_O < V_{CC} + 0.5V ..... ±25mA
DC V_{CC} or Ground Current, I_{CC} or I_{GND} ..... ±50mA

Operating Conditions

Temperature Range (T_A) ..... -55°C to 125°C
Supply Voltage Range, V_{CC}
HC Types ..... 2V to 6V
HCT Types ..... 4.5V to 5.5V
DC Input or Output Voltage, V_I, V_O ..... 0V to V_{CC}
Input Rise and Fall Time
2V ..... 1000ns (Max)
4.5V ..... 500ns (Max)
6V ..... 400ns (Max)

Thermal Information

Thermal Resistance (Typical, Note 3)	θ_{JA} (°C/W)
PDIP Package	90
SOIC Package	115

Maximum Junction Temperature ..... 150°C
Maximum Storage Temperature Range ..... -65°C to 150°C
Maximum Lead Temperature (Soldering 10s) ..... 300°C
(SOIC - Lead Tips Only)

CAUTION: Stresses above those listed in "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

NOTE:

3. θ_{JA} is measured with the component mounted on an evaluation PC board in free air.

DC Electrical Specifications

PARAMETER	SYMBOL	TEST CONDITIONS V_I (V)	TEST CONDITIONS I_O (mA)	V_{CC} (V)	25°C MIN	25°C TYP	25°C MAX	-40°C TO 85°C MIN	-40°C TO 85°C MAX	-55°C TO 125°C MIN	-55°C TO 125°C MAX	UNITS
HC TYPES												
High Level Input Voltage	V_{IH}	-	-	2	1.5	-	-	1.5	-	1.5	-	V
				4.5	3.15	-	-	3.15	-	3.15	-	V
				6	4.2	-	-	4.2	-	4.2	-	V
Low Level Input Voltage	V_{IL}	-	-	2	-	-	0.5	-	0.5	-	0.5	V
				4.5	-	-	1.35	-	1.35	-	1.35	V
				6	-	-	1.8	-	1.8	-	1.8	V
High Level Output Voltage CMOS Loads	V_{OH}	V_{IH} or V_{IL}	-0.02	2	1.9	-	-	1.9	-	1.9	-	V
			-0.02	4.5	4.4	-	-	4.4	-	4.4	-	V
			-0.02	6	5.9	-	-	5.9	-	5.9	-	V
High Level Output Voltage TTL Loads			-4	4.5	3.98	-	-	3.84	-	3.7	-	V
			-5.2	6	5.48	-	-	5.34	-	5.2	-	V
Low Level Output Voltage CMOS Loads	V_{OL}	V_{IH} or V_{IL}	0.02	2	-	-	0.1	-	0.1	-	0.1	V
			0.02	4.5	-	-	0.1	-	0.1	-	0.1	V
			0.02	6	-	-	0.1	-	0.1	-	0.1	V
Low Level Output Voltage TTL Loads			4	4.5	-	-	0.26	-	0.33	-	0.4	V
			5.2	6	-	-	0.26	-	0.33	-	0.4	V
Input Leakage Current	I_I	V_{CC} or GND	-	6	-	-	±0.1	-	±1	-	±1	µA

CD74HC165, CD74HCT165

DC Electrical Specifications (Continued)

PARAMETER	SYMBOL	TEST CONDITIONS V_I (V)	TEST CONDITIONS I_O (mA)	V_{CC} (V)	25°C MIN	25°C TYP	25°C MAX	-40°C TO 85°C MIN	-40°C TO 85°C MAX	-55°C TO 125°C MIN	-55°C TO 125°C MAX	UNITS
Quiescent Device Current	I_{CC}	V_{CC} or GND	0	6	-	-	8	-	80	-	160	µA
HCT TYPES												
High Level Input Voltage	V_{IH}	-	-	4.5 to 5.5	2	-	-	2	-	2	-	V
Low Level Input Voltage	V_{IL}	-	-	4.5 to 5.5	-	-	0.8	-	0.8	-	0.8	V
High Level Output Voltage CMOS Loads	V_{OH}	V_{IH} or V_{IL}	-0.02	4.5	4.4	-	-	4.4	-	4.4	-	V
High Level Output Voltage TTL Loads			-4	4.5	3.98	-	-	3.84	-	3.7	-	V
Low Level Output Voltage CMOS Loads	V_{OL}	V_{IH} or V_{IL}	0.02	4.5	-	-	0.1	-	0.1	-	0.1	V
Low Level Output Voltage TTL Loads			4	4.5	-	-	0.26	-	0.33	-	0.4	V
Input Leakage Current	I_I	V_{CC} to GND	0	5.5	-	-	±0.1	-	±1	-	±1	µA
Quiescent Device Current	I_{CC}	V_{CC} or GND	0	5.5	-	-	8	-	80	-	160	µA
Additional Quiescent Device Current Per Input Pin: 1 Unit Load (Note 4)	ΔI_{CC} (Note 4)	V_{CC} -2.1	-	4.5 to 5.5	-	100	360	-	450	-	490	µA

NOTE:

4. For dual-supply systems theoretical worst case (V_I = 2.4V, V_{CC} = 5.5V) specification is 1.8mA.

HCT Input Loading Table

INPUT	UNIT LOADS
DS, D0 to D7	0.35
CP, $\overline{PL}$	0.65

NOTE: Unit Load is ΔI_{CC} limit specified in DC Electrical Specifications table, e.g. 360µA max at 25°C.

Prerequisite For Switching Specifications

PARAMETER	SYMBOL	V_{CC} (V)	25°C MIN	25°C MAX	-40°C TO 85°C MIN	-40°C TO 85°C MAX	-55°C TO 125°C MIN	-55°C TO 125°C MAX	UNITS
HC TYPES									
CP Pulse Width	t_{WL}, t_{WH}	2	80	-	100	-	120	-	ns
		4.5	16	-	20	-	24	-	ns
		6	14	-	17	-	20	-	ns

CD74HC165, CD74HCT165

Prerequisite For Switching Specifications (Continued)

PARAMETER	SYMBOL	V_{CC} (V)	25°C MIN	25°C MAX	-40°C TO 85°C MIN	-40°C TO 85°C MAX	-55°C TO 125°C MIN	-55°C TO 125°C MAX	UNITS
$\overline{PL}$ Pulse Width	t_{WL}	2	80	-	100	-	120	-	ns
		4.5	16	-	20	-	24	-	ns
		6	14	-	17	-	20	-	ns
Set-up Time	t_{SU}	2	80	-	100	-	120	-	ns
DS to CP		4.5	16	-	20	-	24	-	ns
		6	14	-	17	-	20	-	ns
$\overline{CE}$ to CP	$t_{SU(L)}$	2	80	-	100	-	120	-	ns
		4.5	16	-	20	-	24	-	ns
		6	14	-	17	-	20	-	ns
D0-D7 to $\overline{PL}$	t_{SU}	2	80	-	100	-	120	-	ns
		4.5	16	-	20	-	24	-	ns
		6	14	-	17	-	20	-	ns
Hold Time	t_H	2	35	-	45	-	55	-	ns
DS to CP or $\overline{CE}$		4.5	7	-	9	-	11	-	ns
		6	6	-	8	-	9	-	ns
$\overline{CE}$ to CP	t_H	2	0	-	0	-	0	-	ns
		4.5	0	-	0	-	0	-	ns
		6	0	-	0	-	0	-	ns
Recovery Time	t_{REC}	2	100	-	125	-	150	-	ns
$\overline{PL}$ to CP		4.5	20	-	25	-	30	-	ns
		6	17	-	21	-	26	-	ns
Maximum Clock Pulse Frequency	f_{MAX}	2	6	-	5	-	4	-	MHz
		4.5	30	-	24	-	20	-	MHz
		6	35	-	28	-	24	-	MHz
HCT TYPES									
CP Pulse Width	t_{WL}, t_{WH}	4.5	18	-	23	-	27	-	ns
$\overline{PL}$ Pulse Width	t_{WL}	4.5	20	-	25	-	30	-	ns
Set-up Time DS to CP	t_{SU}	4.5	20	-	25	-	30	-	ns
$\overline{CE}$ to CP	$t_{SU(L)}$	4.5	20	-	25	-	30	-	ns
D0-D7 to $\overline{PL}$	t_{SU}	6	20	-	25	-	30	-	ns
Hold Time DS to CP or $\overline{CE}$	t_H	4.5	7	-	9	-	11	-	ns
$\overline{CE}$ to CP	t_S, t_H	4.5	0	-	0	-	0	-	ns
Recovery Time $\overline{PL}$ to CP	t_{REC}	4.5	20	-	25	-	30	-	ns
Maximum Clock Pulse Frequency	f_{MAX}	4.5	27	-	22	-	18	-	MHz

CD74HC165, CD74HCT165

Switching Specifications Input t_r, t_f = 6ns

PARAMETER	SYMBOL	TEST CONDITIONS	V_{CC} (V)	25°C TYP	25°C MAX	-40°C TO 85°C MAX	-55°C TO 125°C MAX	UNITS
HC TYPES								
Propagation Delay	t_{PLH}, t_{PHL}	C_L = 50pF	2	-	165	205	250	ns
CP or $\overline{CE}$ to Q_7 or $\overline{Q_7}$			4.5	-	33	41	50	ns
		C_L = 15pF	5	13	-	-	-	ns
		C_L = 50pF	6	-	28	35	43	ns
$\overline{PL}$ to Q_7 or $\overline{Q_7}$	t_{PLH}, t_{PHL}	C_L = 50pF	2	-	175	220	265	ns
			4.5	-	35	44	53	ns
		C_L = 15pF	5	14	-	-	-	ns
		C_L = 50pF	6	-	30	37	45	ns
D7 to Q_7 or $\overline{Q_7}$	t_{PLH}, t_{PHL}	C_L = 50pF	2	-	150	190	225	ns
			4.5	-	30	38	45	ns
		C_L = 15pF	5	12	-	-	-	ns
		C_L = 50pF	6	-	26	33	38	ns
Output Transition Times	t_{TLH}, t_{THL}	C_L = 50pF	2	-	75	95	110	ns
			4.5	-	15	19	22	ns
			6	-	13	16	19	ns
Input Capacitance	C_{IN}	-	-	-	10	10	10	pF
Power Dissipation Capacitance (Notes 5, 6)	C_{PD}	-	5	17	-	-	-	pF
HCT TYPES								
Propagation Delay	t_{PLH}, t_{PHL}	C_L = 50pF	4.5	-	40	50	60	ns
CP or $\overline{CE}$ to Q_7 or $\overline{Q_7}$		C_L = 15pF	5	17	-	-	-	ns
$\overline{PL}$ to Q_7 or $\overline{Q_7}$	t_{PLH}, t_{PHL}	C_L = 50pF	4.5	-	40	50	60	ns
		C_L = 15pF	5	17	-	-	-	ns
D7 to Q_7 or $\overline{Q_7}$	t_{PLH}, t_{PHL}	C_L = 50pF	4.5	-	35	44	53	ns
		C_L = 15pF	5	14	-	-	-	ns
Output Transition Times	t_{TLH}, t_{THL}	C_L = 50pF	4.5	-	15	19	22	ns
Input Capacitance	C_{IN}	C_L = 50pF	-	-	10	10	10	pF
Power Dissipation Capacitance (Notes 5, 6)	C_{PD}	-	5	24		-	-	pF

NOTES:

5. C_{PD} is used to determine the dynamic power consumption, per package.
6. $P_D = V_{CC}^2 f_i + \Sigma (C_L V_{CC}^2 + f_O)$ where f_i = Input Frequency, f_O = Output Frequency, C_L = Output Load Capacitance, V_{CC} = Supply Voltage.

Test Circuits and Waveforms

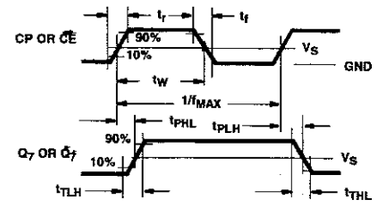


FIGURE 3. SERIAL-SHIFT MODE

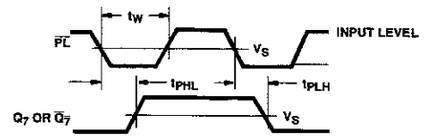


FIGURE 4. PARALLEL-LOAD MODE

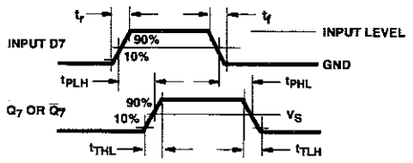


FIGURE 5. PARALLEL-LOAD MODE

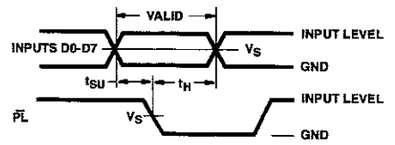


FIGURE 6. PARALLEL-LOAD MODE

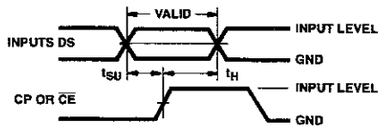


FIGURE 7. SERIAL-SHIFT MODE

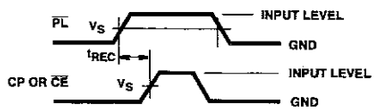


FIGURE 8. SERIAL-SHIFT MODE

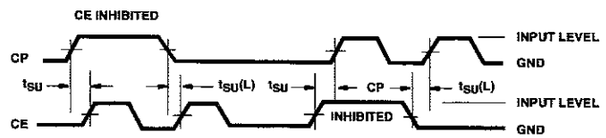


FIGURE 9. SERIAL-SHIFT, CLOCK-INHIBIT MODE

APÉNDICE B

CONVERSIONES

Decimal	BCD(8421)	Octal	Binario	Decimal	BCD(8421)	Octal	Binario	Decimal	BCD(8421)	Octal	Binario
0	0000	0	0	34	00110100	42	100010	68	01101000	104	1000100
1	0001	1	1	35	00110101	43	100011	69	01101001	105	1000101
2	0010	2	10	36	00110110	44	100100	70	01110000	106	1000110
3	0011	3	11	37	00110111	45	100101	71	01110001	107	1000111
4	0100	4	100	38	00111000	46	100110	72	01110010	110	1001000
5	0101	5	101	39	00111001	47	100111	73	01110011	111	1001001
6	0110	6	110	40	01000000	50	101000	74	01110100	112	1001010
7	0111	7	111	41	01000001	51	101001	75	01110101	113	1001011
8	1000	10	1000	42	01000010	52	101010	76	01110110	114	1001100
9	1001	11	1001	43	01000011	53	101011	77	01110111	115	1001101
10	00010000	12	1010	44	01000100	54	101100	78	01111000	116	1001110
11	00010001	13	1011	45	01000101	55	101101	79	01111001	117	1001111
12	00010010	14	1100	46	01000110	56	101110	80	10000000	120	1010000
13	00010011	15	1101	47	01000111	57	101111	81	10000001	121	1010001
14	00010100	16	1110	48	01001000	60	110000	82	10000010	122	1010010
15	00010101	17	1111	49	01001001	61	110001	83	10000011	123	1010011
16	00010110	20	10000	50	01010000	62	110010	84	10000100	124	1010100
17	00010111	21	10001	51	01010001	63	110011	85	10000101	125	1010101
18	00011000	22	10010	52	01010010	64	110100	86	10000110	126	1010110
19	00011001	23	10011	53	01010011	65	110101	87	10000111	127	1010111
20	00100000	24	10100	54	01010100	66	110110	88	10001000	130	1011000
21	00100001	25	10101	55	01010101	67	110111	89	10001001	131	1011001
22	00100010	26	10110	56	01010110	70	111000	90	10010000	132	1011010
23	00100011	27	10111	57	01010111	71	111001	91	10010001	133	1011011
24	00100100	30	11000	58	01011000	72	111010	92	10010010	134	1011100
25	00100101	31	11001	59	01011001	73	111011	93	10010011	135	1011101
26	00100110	32	11010	60	01100000	74	111100	94	10010100	136	1011110
27	00100111	33	11011	61	01100001	75	111101	95	10010101	137	1011111
28	00101000	34	11100	62	01100010	76	111110	96	10010110	140	1100000
29	00101001	35	11101	63	01100011	77	111111	97	10010111	141	1100001
30	00110000	36	11110	64	01100100	100	1000000	98	10011000	142	1100010
31	00110001	37	11111	65	01100101	101	1000001	99	10011001	143	1100011
32	00110010	40	100000	66	01100110	102	1000010				
33	00110011	41	100001	67	01100111	103	1000011				

Potencias de dos

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125
1 099 511 627 776	40	0.000 000 000 000 909 494 701 772 928 237 915 039 062 5
2 199 023 255 552	41	0.000 000 000 000 454 747 350 886 464 118 957 519 531 25
4 398 046 511 104	42	0.000 000 000 000 227 373 675 443 232 059 478 759 765 625
8 796 093 022 208	43	0.000 000 000 000 113 686 837 721 616 029 739 379 882 812 5
17 592 186 044 416	44	0.000 000 000 000 056 843 418 860 808 014 869 689 941 406 25
35 184 372 088 832	45	0.000 000 000 000 028 421 709 430 404 007 434 844 970 703 125
70 368 744 177 664	46	0.000 000 000 000 014 210 854 715 202 003 717 422 485 351 562 5
140 737 488 355 328	47	0.000 000 000 000 007 105 427 357 601 001 858 711 242 675 781 25
281 474 976 710 656	48	0.000 000 000 000 003 552 713 678 800 500 929 355 621 337 890 625
562 949 953 421 312	49	0.000 000 000 000 001 776 356 839 400 250 464 677 810 668 945 312 5
1 125 899 906 842 624	50	0.000 000 000 000 000 888 178 419 700 125 232 338 905 334 472 656 25
2 251 799 813 685 248	51	0.000 000 000 000 000 444 089 209 850 062 616 169 452 667 236 328 125
4 503 599 627 370 496	52	0.000 000 000 000 000 222 044 604 925 031 308 084 726 333 618 164 062 5
9 007 199 254 740 992	53	0.000 000 000 000 000 111 022 302 462 515 654 042 363 166 809 082 031 25
18 014 398 509 481 984	54	0.000 000 000 000 000 055 511 151 231 257 827 021 181 583 404 541 015 625
36 028 797 018 963 968	55	0.000 000 000 000 000 027 755 575 615 628 913 510 590 791 702 270 507 812 5
72 057 594 037 927 936	56	0.000 000 000 000 000 013 877 787 807 814 456 755 295 395 851 135 253 906 25
144 115 188 075 855 872	57	0.000 000 000 000 000 006 938 893 903 907 228 377 647 697 925 567 626 953 125
288 230 376 151 711 744	58	0.000 000 000 000 000 003 469 446 951 953 614 188 823 848 962 783 813 476 562 5
576 460 752 303 423 488	59	0.000 000 000 000 000 001 734 723 475 976 807 094 411 924 481 391 906 738 281 25
1 152 921 504 606 846 976	60	0.000 000 000 000 000 000 867 361 737 988 403 547 205 962 240 695 953 369 140 625
2 305 843 009 213 693 952	61	0.000 000 000 000 000 000 433 680 868 994 201 773 602 981 120 347 976 684 570 312 5
4 611 686 018 427 387 904	62	0.000 000 000 000 000 000 216 840 434 497 100 886 801 490 560 173 988 342 285 156 25
9 223 372 036 854 775 808	63	0.000 000 000 000 000 000 108 420 217 248 550 443 400 745 280 086 994 171 142 578 125
18 446 744 073 709 551 616	64	0.000 000 000 000 000 000 054 210 108 624 275 221 700 372 640 043 497 085 571 289 062 5
36 893 488 147 419 103 232	65	0.000 000 000 000 000 000 027 105 054 312 137 610 850 186 320 021 748 542 785 644 531 25
73 786 976 294 838 206 464	66	0.000 000 000 000 000 000 013 552 527 156 088 805 425 093 160 010 874 271 392 822 265 625
147 573 952 589 676 412 928	67	0.000 000 000 000 000 000 006 776 263 578 034 402 712 546 580 005 437 135 696 411 132 812 5
295 147 905 179 352 825 856	68	0.000 000 000 000 000 000 003 388 131 789 017 201 356 273 290 002 718 567 848 205 566 406 25
590 295 810 358 705 651 712	69	0.000 000 000 000 000 000 001 694 065 894 508 600 678 136 645 001 359 283 924 102 783 203 125
1 180 591 620 717 411 303 424	70	0.000 000 000 000 000 000 000 847 032 947 254 300 339 068 322 500 679 641 962 051 391 601 562 5
2 361 183 241 434 822 606 848	71	0.000 000 000 000 000 000 000 423 516 473 627 150 169 534 161 250 339 820 981 025 695 800 781 25
4 722 366 482 869 645 213 696	72	0.000 000 000 000 000 000 000 211 758 236 813 575 084 767 080 625 169 910 490 512 847 900 390 625

APÉNDICE C

■ SUMADORES CON ACARREO SERIE VERSUS ACARREO ANTICIPADO

Sumadores con acarreo serie

Un **sumador con acarreo serie** es aquél en que la salida de acarreo de cada sumador completo se conecta a la entrada de acarreo de la siguiente etapa (una etapa es un sumador completo). La suma y el acarreo de salida de cualquier etapa no se generan hasta que se introduce el acarreo de entrada. Esto origina un cierto retraso en el proceso de adición, como se muestra en la Figura C.1. El retardo de propagación del acarreo para cada sumador completo es el tiempo que transcurre desde que se introduce el acarreo de entrada hasta que se genera el acarreo de salida, suponiendo que las dos entradas *A* y *B* ya están presentes en el sumador.

El sumador completo 1 (SC1) no puede producir un acarreo de salida hasta que se introduce el acarreo de entrada. El sumador completo 2 (SC2) no puede generar un acarreo de salida hasta que no lo haga SC1. El sumador completo 3 (SC3) no produce un acarreo de salida hasta que no lo produzcan SC1 y SC2, y así sucesivamente. Como puede verse en la Figura C.1, el acarreo de entrada de la etapa menos significativa tiene que propagarse a través de todos los sumadores hasta que se genere la suma final. El retardo acumulado a través de todas las etapas de sumadores es el tiempo que tarda en calcularse la suma en el caso peor. El retardo total puede variar dependiendo del bit de acarreo que cada sumador completo produce. Si se suman dos números que no generen acarreo (0) entre cada etapa, el tiempo de suma es sencillamente el tiempo de propagación a través de un único sumador completo, desde que se aplican los bits de datos a las entradas hasta que se obtiene la suma.

Sumadores con acarreo anticipado

La velocidad con la que puede realizarse una adición se encuentra limitada por el tiempo requerido por los acarreos para propagarse a través de cada etapa de un sumador en paralelo. Una manera de acelerar el proceso de adición consiste en eliminar este retardo de propagación del acarreo, técnica que recibe el nombre de adición con **acarreo anticipado.** Los sumadores con acarreo anticipado adelantan el acarreo de salida de cada etapa, y basándose en los valores de las entradas del sumador producen el acarreo de salida por generación o por propagación de acarreo.

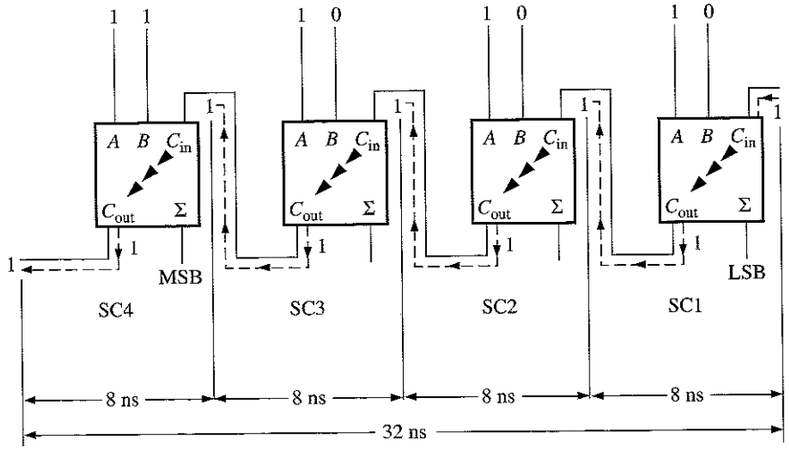


FIGURA C.1
Sumador con acarreo serie de 4 bits, mostrando el caso más desfavorable de retraso en la propagación del acarreo.

La **generación del acarreo** ocurre cuando el sumador completo genera internamente el acarreo de salida. Se genera un acarreo sólo cuando ambos bits de entrada son 1. Este acarreo generado C_g se expresa como la función AND entre los dos bits de entrada, *A* y *B*.

$$C_g = AB \tag{C.1}$$

La **propagación del acarreo** ocurre cuando el acarreo de entrada se transmite como acarreo de salida. Un acarreo de entrada puede ser propagado por el sumador completo cuando al menos uno de los bits de entrada valga 1. El acarreo serie, C_p, se expresa como la función OR de los bits de entrada.

$$C_p = A + B \tag{C.2}$$

Las condiciones para generación y propagación del acarreo se muestran en la Figura C.2. Las tres flechas representan la propagación.

El acarreo de salida de un sumador completo se puede expresar en función del acarreo generado (C_g) y del acarreo serie (C_p). El acarreo de salida (C_{out}) vale 1 si el acarreo generado es 1, o si el acarreo serie es 1 y el acarreo de entrada (C_{in}) también es 1. En otras palabras, obtenemos un acarreo de salida igual a 1 si el sumador completo (*A*=1 y *B*=1) lo genera, o si el sumador propaga el acarreo de entrada [(*A*=1 o *B*=1) y C_{in}=1]. Esta relación se expresa como:

$$C_{out} = C_g + C_p C_{in} \tag{C.3}$$

Vamos a ver a continuación cómo se aplican estos conceptos a un sumador en paralelo, cuyas etapas individuales se muestran en la Figura C.3 para un ejemplo de 4 bits. Para cada sumador completo, el acarreo de salida depende del acarreo generado

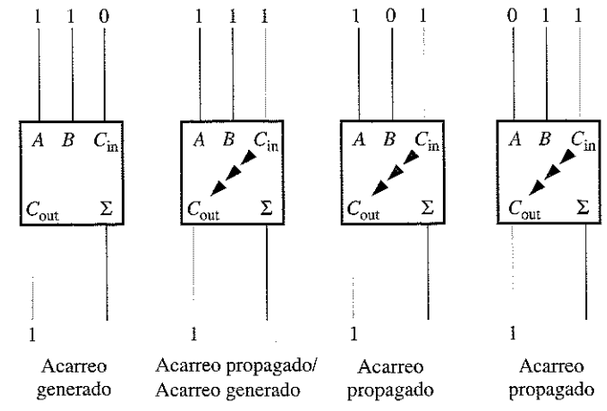


FIGURA C.2
Condiciones para la generación y propagación del acarreo.

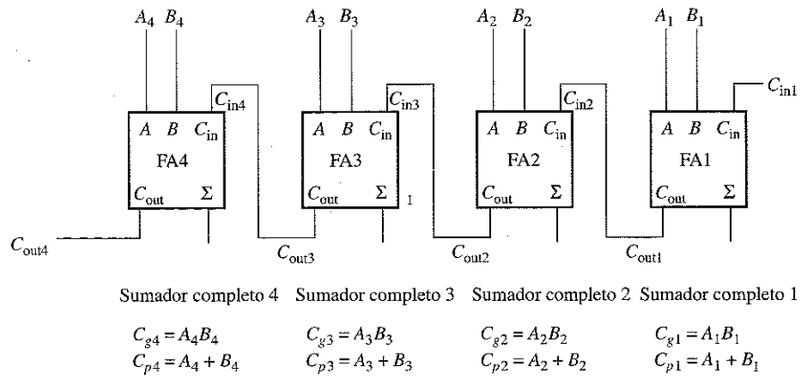


FIGURA C.3
Generación y propagación del acarreo en función de los bits de entrada en un sumador de 4 bits.

(C_g), del acarreo serie (C_p) y del acarreo de entrada (C_{in}). Las funciones C_g y C_p de cada etapa se encuentran *inmediatamente* disponibles tan pronto como se aplican las entradas y el acarreo de entrada al sumador menos significativo, ya que dependen sólo de estos bits. El acarreo de entrada de cada etapa es el acarreo de salida de la etapa inmediatamente anterior.

Basándonos en este análisis, se pueden desarrollar las expresiones para el acarreo de salida, C_{out}, de cada sumador completo para este ejemplo de 4 bits.

Sumador completo 1:

$$C_{out1} = C_{g1} + C_{p1}C_{in1} \tag{C.4}$$

Sumador completo 2:

$$\begin{aligned} C_{in2} &= C_{out1} \\ C_{out2} &= C_{g2} + C_{p2}C_{in2} \\ &= C_{g2} + C_{p2}C_{out1} \\ &= C_{g2} + C_{p2}(C_{g1} + C_{p1}C_{in1}) \end{aligned}$$

$$C_{out2} = C_{g2} + C_{p2}C_{g1} + C_{p2}C_{p1}C_{in1} \tag{C.5}$$

Sumador completo 3:

$$\begin{aligned} C_{in3} &= C_{out2} \\ C_{out3} &= C_{g3} + C_{p3}C_{in3} \\ &= C_{g3} + C_{p3}C_{out2} \\ &= C_{g3} + C_{p3}(C_{g2} + C_{p2}C_{g1} + C_{p2}C_{p1}C_{in1}) \end{aligned}$$

$$C_{out3} = C_{g3} + C_{p3}C_{g2} + C_{p3}C_{p2}C_{g1} + C_{p3}C_{p2}C_{p1}C_{in1} \tag{C.6}$$

Sumador completo 4:

$$\begin{aligned} C_{in4} &= C_{out3} \\ C_{out4} &= C_{g4} + C_{p4}C_{in4} \end{aligned}$$

$$\begin{aligned} &= C_{g4} + C_{p4}C_{out3} \\ &= C_{g4} + C_{p4}(C_{g3} + C_{p3}C_{g2} + C_{p3}C_{p2}C_{g1} + C_{p3}C_{p2}C_{p1}C_{in1}) \end{aligned}$$

$$C_{out4} = C_{g4} + C_{p4}C_{g3} + C_{p4}C_{p3}C_{g2} + C_{p4}C_{p3}C_{p2}C_{g1} + C_{p4}C_{p3}C_{p2}C_{p1}C_{in1} \tag{C.7}$$

Observe que, en cada una de estas expresiones, el acarreo de salida de cada sumador completo depende únicamente del acarreo de entrada inicial, C_{in1}, de las funciones C_g y C_p en cada etapa y de las funciones C_g y C_p de las etapas precedentes. Dado que cada una de las funciones C_g y C_p se pueden expresar en función de las entradas del sumador, A y B, todos los acarreos de salida se encuentran inmediatamente disponibles (excepto por el retardo de las puertas) y no se tiene que esperar a que se propague el acarreo a través de todas las etapas para obtener el resultado final. Por consiguiente, el proceso de acarreo anticipado aumenta la velocidad de un proceso de adición.

Las ecuaciones C.4 a la C.7 se implementan mediante puertas lógicas y se conectan a los sumadores completos para conseguir un sumador de cuatro bits con acarreo anticipado, como muestra la Figura C.4.

Combinación de sumadores con acarreo anticipado y serie

Los sumadores 74LS83A y 74LS283 de 4 bits son sumadores con acarreo anticipado. Cuando estos sumadores se conectan en cascada para ampliar su capacidad de manejo de números binarios de más de cuatro bits, el acarreo de salida de uno de los sumadores se conecta al acarreo de entrada del siguiente. Esto origina una condición de acarreo serie entre estos sumadores, de forma que cuando dos o más 74LS283A o 74LS283 se conectan en cascada, el sumador resultante es en realidad una combinación de sumadores con acarreo serie y anticipado. La función del acarreo anticipado es un proceso interno de cada sumador MSI, y la característica de acarreo serie se produce cuando se conectan entre sí los acarreos de salida y los de entrada de las distintas etapas del circuito.

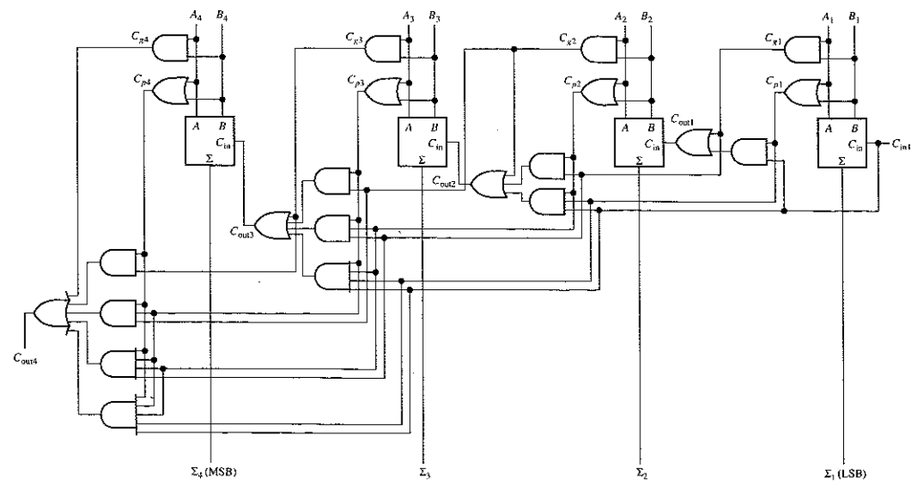


FIGURA C.4
Diagrama lógico de un sumador de cuatro etapas con acarreo anticipado.

RESPUESTAS A LOS PROBLEMAS IMPARES

CAPÍTULO 1

1. Los datos digitales se pueden transmitir y almacenar de forma más eficiente y fiable.

3. **(a)** 11010001 **(b)** 000101010

5. **(a)** 550 ns **(b)** 600 ns **(c)** 2,7 μs **(d)** 10 V

7. 250 Hz

9. 50%

11. 8 μs; 1μs

13. Puerta AND

15. **(a)** sumador **(b)** multiplicador **(c)** multiplexor **(d)** comparador

17. 01010000

19. Los pines de los componentes DIP se insertan en los taladros de la tarjeta de circuito impreso. Los pines de los componentes SMT se conectan a los pads superficiales.

21. 7 V.

23. Un conjunto de circuitos interconectados para realizar una función específica.

25. Introducir un nuevo valor a través del teclado numérico.

CAPÍTULO 2

1. **(a)** 1 **(b)** 100 **(c)** 100.000

3. **(a)** 400; 70; 1 **(b)** 9000; 300; 50; 6 **(c)** 100.000; 20.000; 5000; 0; 0; 0

5. **(a)** 3 **(b)** 4 **(c)** 7 **(d)** 8 **(e)** 9 **(f)** 12 **(g)** 11 **(h)** 15

7. **(a)** 51,75 **(b)** 42,25 **(c)** 65,875 **(d)** 120,625
(e) 92,65625 **(f)** 113,0625 **(g)** 90,625 **(h)** 127,96875

9. **(a)** 5 bits **(b)** 6 bits **(c)** 6 bits **(d)** 7 bits
(e) 7 bits **(f)** 7 bits **(g)** 8 bits **(h)** 8 bits

11. **(a)** 1010 **(b)** 10001 **(c)** 11000 **(d)** 110000
(e) 111101 **(f)** 1011101 **(g)** 1111101 **(h)** 10111010

13. **(a)** 1111 **(b)** 10101 **(c)** 11100 **(d)** 100010
(e) 101000 **(f)** 111011 **(g)** 1000001 **(h)** 1001001

15. **(a)** 100 **(b)** 100 **(c)** 1000 **(d)** 1101 **(e)** 1110 **(f)** 11000

17. **(a)** 1001 **(b)** 1000 **(c)** 100011
(d) 110110 **(e)** 10101001 **(f)** 10110110

19. **(a)** 010 **(b)** 001 **(c)** 0101 **(d)** 00101000 **(e)** 0001010 **(f)** 11110

21. **(a)** 00011101 **(b)** 11010101 **(c)** 01100100 **(d)** 11111011

23. **(a)** 00001100 **(b)** 10111100 **(c)** 01100101 **(d)** 10000011

25. **(a)** −102 **(b)** +116 **(c)** −64

27. **(a)** 0 10001101 11110000101011000000000
(b) 1 10001010 10000011000000000000000

29. **(a)** 00110000 **(b)** 00011101 **(c)** 11101011 **(d)** 100111110

31. **(a)** 11000101 **(b)** 11000000

33. 100111001010

35. **(a)** 00111000 **(b)** 01011001 **(c)** 101000010100 **(d)** 010111001000
(e) 0100000100000000 **(f)** 1111101100010111 **(g)** 1000101010011101

37. **(a)** 35 **(b)** 146 **(c)** 26 **(d)** 141
(e) 243 **(f)** 235 **(g)** 1474 **(h)** 1792

39. **(a)** 60_{16} **(b)** $10B_{16}$ **(c)** $1BA_{16}$

41. **(a)** 10 **(b)** 23 **(c)** 46 **(d)** 52 **(e)** 67
(f) 367 **(g)** 115 **(h)** 532 **(i)** 4085

43. **(a)** 001011 **(b)** 101111 **(c)** 001000001
(d) 011010001 **(e)** 101100000 **(f)** 100110101011
(g) 001011010111001 **(h)** 100101110000000 **(i)** 001000000010001011

45. **(a)** 00010000 **(b)** 00010011 **(c)** 00011000 **(d)** 00100001
(e) 00100101 **(f)** 00110110 **(g)** 01000100 **(h)** 01010111
(i) 01101001 **(j)** 10011000 **(k)** 000100100101 **(l)** 000101010110

47. **(a)** 000100000100 **(b)** 000100101000 **(c)** 000100110010
(d) 000101010000 **(e)** 000110000110 **(f)** 001000010000
(g) 001101011001 **(h)** 010101000111 **(i)** 0001000001010001

49. **(a)** 80 **(b)** 237 **(c)** 346 **(d)** 421 **(e)** 754
(f) 800 **(g)** 978 **(h)** 1683 **(i)** 9018 **(j)** 6667

51. **(a)** 00010100 **(b)** 00010010 **(c)** 00010111 **(d)** 00010110
(e) 01010010 **(f)** 000100001001 **(g)** 000110010101 **(h)** 0001001001101001

53. El código Gray hace que sólo cambie un bit cada vez cuando pasa de un número de la secuencia al siguiente.

55. **(a)** 1100 **(b)** 00011 **(c)** 10000011110

57. **(a)** CAN **(b)** J **(c)** = **(d)** # **(e)** > **(f)** B

59. 48 65 6C 6C 6F 2E 20 48 6F 77 20 61 72 65 20 79 6F 75 3F

61. **(b)** Es incorrecto.

63. **(a)** 110100100 **(b)** 000001001 **(c)** 111111110

65. **(a)** 00100101 (25 BCD) **(b)** 00011001 (25 en binario)
(c) 00001111 (15) **(d)** 0110000110010010 (24.975)
(e) 0110000110100001 (24.990)

CAPÍTULO 3

1. Véase la Figura P.1

3. Véase la Figura P.2

5. Véase la Figura P.3

7. Véase la Figura P.4

9. Véase la Figura P.5

11. Véase la Figura P.6

FIGURA P.1

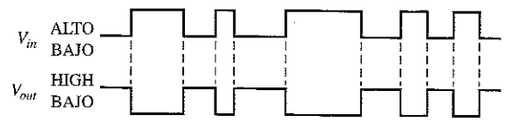


FIGURA P.2

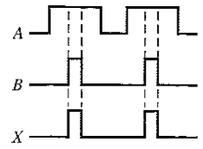


FIGURA P.3

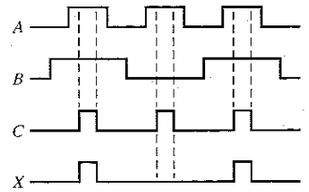


FIGURA P.4

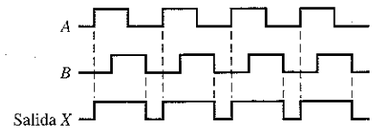


FIGURA P.5

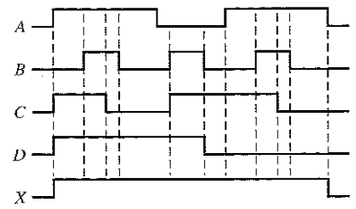


FIGURA P.6

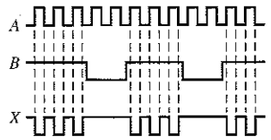


FIGURA P.7

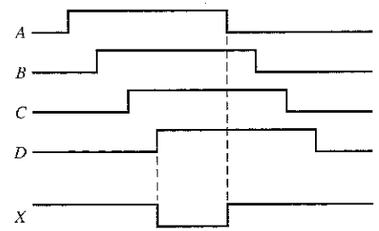


FIGURA P.8

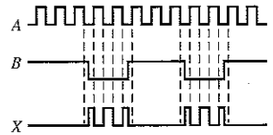
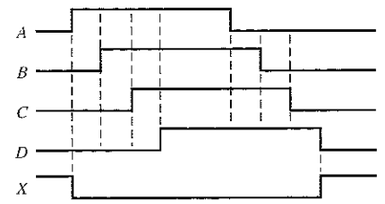


FIGURA P.9



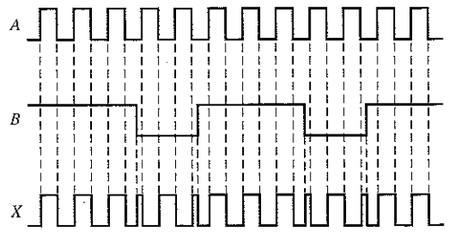


FIGURA P.10

13. Véase la Figura P.7

15. Véase la Figura P.8

17. Véase la Figura P.9

19. XOR = $A\overline{B} + \overline{A}B$; OR = $A + B$

21. Véase la Figura P.10

23. CMOS

25. $t_{PLH} = 4{,}3$ ns; $t_{PHL} = 10{,}5$ ns

27. 20 mW

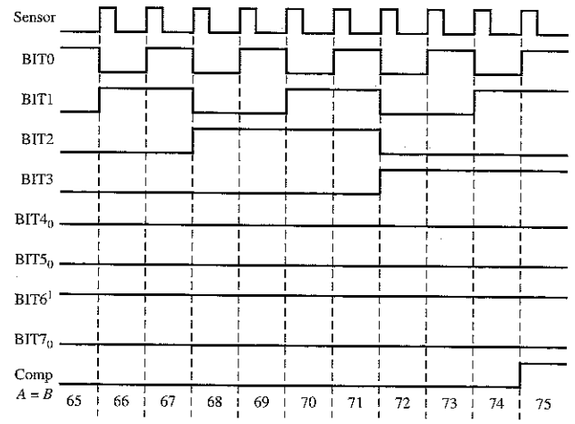
29. Las puertas de los apartados (b), (c) y (e) fallan

31. **(a)** Salida defectuosa (está en circuito abierto o se mantiene a nivel BAJO)
(b) La entrada del pin 4 o la salida del pin 6 están internamente en circuito abierto.

33. La entrada de cinturón de seguridad a la puerta AND está en circuito abierto.

35. Véase la Figura P.11.

FIGURA P.11



37. La entrada BIT5 permanentemente a 1 o en circuito abierto podría producirse por una entrada en circuito abierto de la correspondiente puerta XOR o la entrada BIT6 se mantiene a 0 debido a un cortocircuito.

39. Véase la Figura P.12.

41. Añadir un inversor a la línea de entrada de habilitación de la puerta AND.

43. Véase la Figura P.13.

45. Las líneas de la válvula y de la cinta transportadora deben tomarse de la salida de la puerta NAND. La línea de registro procede del inversor.

47. Las entradas de puerta están cortocircuitadas.

49. La salida de la puerta está en circuito abierto.

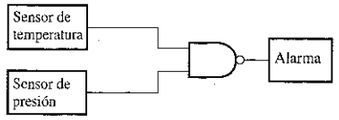


FIGURA P.12

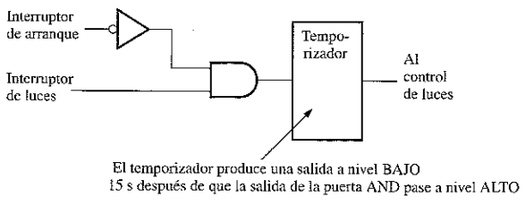


FIGURA P.13

CAPÍTULO 4

1. $X = A + B + C + D$ **3.** $X = \overline{A} + \overline{B} + \overline{C}$

5. **(a)** $AB = 1$ cuando $A = 1, B = 1$
(b) $A\overline{B}C = 1$ cuando $A = 1, B = 0, C = 1$
(c) $A + B = 0$ cuando $A = 0, B = 0$
(d) $\overline{A} + B + \overline{C} = 0$ cuando $A = 1, B = 0, C = 1$
(e) $\overline{A} + \overline{B} + C = 0$ cuando $A = 1, B = 1, C = 0$
(f) $\overline{A} + B = 0$ cuando $A = 1, B = 0$
(g) $A\overline{B}\,\overline{C} = 1$ cuando $A = 1, B = 0, C = 0$

7. **(a)** Conmutativa **(b)** Conmutativa **(c)** Distributiva

9. **(a)** $\overline{A}B$ **(b)** $A + \overline{B}$ **(c)** $\overline{A}\,\overline{B}\,\overline{C}$ **(d)** $\overline{A} + \overline{B} + \overline{C}$
(e) $\overline{A} + \overline{B}\,\overline{C}$ **(f)** $\overline{A} + \overline{B} + \overline{C} + \overline{D}$ **(g)** $(\overline{A} + \overline{B})(\overline{C} + \overline{D})$ **(h)** $\overline{A}B + C\overline{D}$

11. **(a)** $(\overline{A} + \overline{B} + \overline{C})(\overline{E} + \overline{F} + \overline{G})(\overline{H} + \overline{I} + \overline{J})(\overline{K} + \overline{L} + \overline{M})$
(b) $\overline{A}B\overline{C}(\overline{C} + \overline{D}) + BC = \overline{A}B\overline{C} + BC$ **(c)** $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}\,\overline{E}\,\overline{F}\,\overline{G}\,\overline{H}$

13. **(a)** $X = ABCD$ **(b)** $X = AB + C$ **(c)** $X = \overline{\overline{A}\overline{B}}$ **(d)** $X = (A + B)C$

15. Véase la Figura P.14.

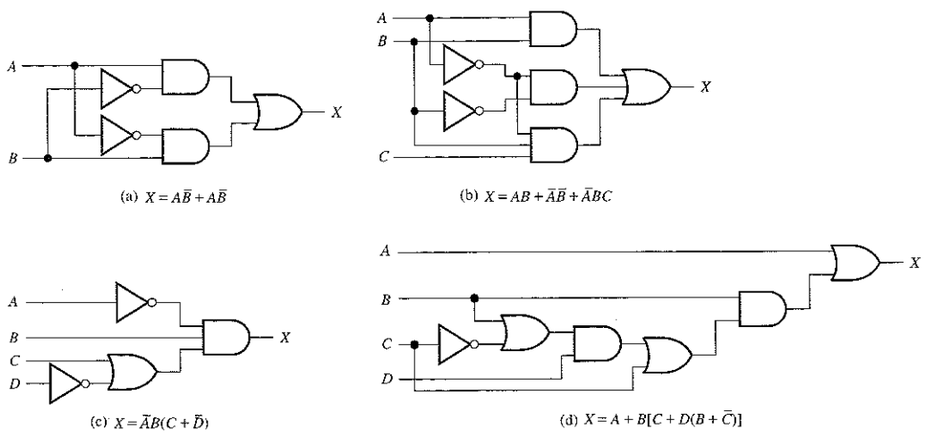


FIGURA P.14

17. **(a)** A **(b)** AB **(c)** C **(d)** A **(e)** $\overline{A}C + \overline{B}C$

19. **(a)** $BD + BE + \overline{D}F$ **(b)** $\overline{A}\,\overline{B}C + \overline{A}\,\overline{B}D$ **(c)** B **(d)** $AB + CD$ **(e)** ABC

21. **(a)** $A\overline{B} + AC + BC$ **(b)** $AC + \overline{B}C$ **(c)** $AB + AC$

23. **(a)** Dominio: A, B, C
Suma de productos estándar: $A\overline{B}C + A\overline{B}\,\overline{C} + ABC + \overline{A}BC$
(b) Dominio: A, B, C
Suma de productos estándar: $ABC + A\overline{B}C + \overline{A}\,\overline{B}C$
(c) Dominio: A, B, C
Suma de productos estándar: $ABC + AB\overline{C} + A\overline{B}C$

25. **(a)** 101 + 100 + 111 + 011 **(b)** 111 + 101 + 001 **(c)** 111 + 110 + 101

27. **(a)** $(A + B + C)(A + B + \overline{C})(A + \overline{B} + C)(\overline{A} + \overline{B} + C)$
(b) $(A + B + C)(A + \overline{B} + C)(A + \overline{B} + \overline{C})(\overline{A} + B + C)(\overline{A} + \overline{B} + C)$
(c) $(A + B + C)(A + B + \overline{C})(A + \overline{B} + C)(A + \overline{B} + \overline{C})(\overline{A} + B + C)$

29. **(a)** Véase la Tabla P.1 **(b)** Véase la Tabla P.2

31. **(a)** Véase la Tabla P.3 **(b)** Véase la Tabla P.4

33. **(a)** Véase la Tabla P.5 **(b)** Véase la Tabla P.6

35. Véase la Figura P.15

37. Véase la Figura P.16

TABLA P.1

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

TABLA P.2

X	Y	Z	Q
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

TABLA P.3

A	B	C	X
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

TABLA P.4

W	X	Y	Z	Q
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

TABLA P.5

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

TABLA P.6

A	B	C	D	X
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

AB \ C	0	1
00	000	001
01	010	011
11	110	111
10	100	101

FIGURA P.15

FIGURA P.16

AB \ C	0	1
00	$\overline{A}\,\overline{B}\,\overline{C}$	$\overline{A}\,\overline{B}C$
01	$\overline{A}B\overline{C}$	$\overline{A}BC$
11	$AB\overline{C}$	ABC
10	$A\overline{B}\,\overline{C}$	$A\overline{B}C$

39. **(a)** No se simplifica **(b)** AC **(c)** $\overline{D}\,\overline{F} + E\overline{F}$

41. **(a)** $AB + AC$
(b) $A + BC$
(c) $B\overline{C}D + A\overline{C}D + BC\overline{D} + AC\overline{D}$
(d) $A\overline{B} + CD$

43. $\overline{B} + C$

45. $\overline{A}\,\overline{B}\,\overline{C}D + C\overline{D} + BC + A\overline{D}$

47 **(a)** $(A + \overline{B} + C + \overline{D})(\overline{A} + B + \overline{C} + D)(\overline{A} + \overline{B} + \overline{C} + \overline{D})$
(b) $(W + \overline{Z})(W + X)(\overline{Y} + \overline{Z})(X + \overline{Y})$

49. $(A + C + D)(A + \overline{B} + C)(\overline{A} + B + \overline{D})(B + \overline{C} + \overline{D})(\overline{A} + \overline{B} + \overline{C} + D)$

51. $X = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}\,\overline{E} + \overline{A}BC\overline{D}\,\overline{E} + AB\overline{C}\,\overline{D}\,\overline{E} + \overline{A}\,\overline{B}DE + \overline{A}BD\overline{E} + \overline{A}\,\overline{C}DE + AB\overline{C}D$

53. LED. Los LED emiten luz, los LCD no.

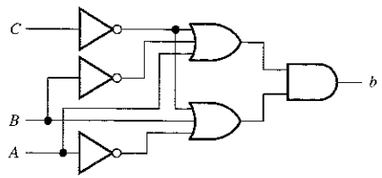
55. Un inversor y seis puertas menos.

57. Añadir un inversor a la salida de la puerta OR de cada uno de los circuitos lógicos de los segmentos.

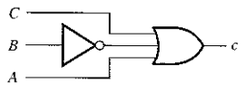
59. Véase la Figura P.17

61. Una entrada invertida o una salida en abierto.

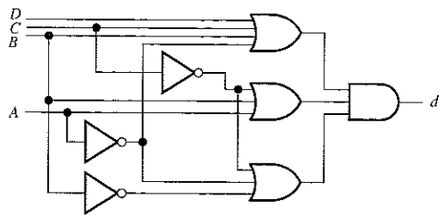
63. La salida de la puerta OR del segmento *b* está en circuito abierto.



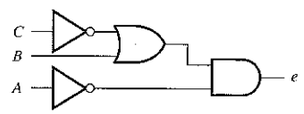
segmento $b = (\overline{C} + B + \overline{A})(\overline{C} + \overline{B} + A)$



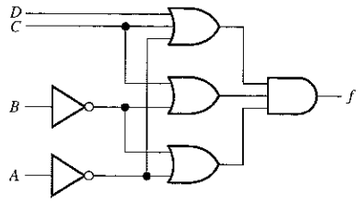
segmento $c = C + \overline{B} + A$



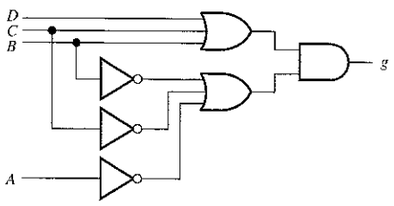
segmento $d = (D + C + B + \overline{A})(\overline{C} + B + A)(\overline{C} + \overline{B} + \overline{A})$



segmento $e = \overline{A}(\overline{C} + B)$



segmento $f = (D + C + \overline{A})(C + \overline{B})(\overline{B} + \overline{A})$



segmento $g = (D + C + B)(\overline{C} + \overline{B} + \overline{A})$

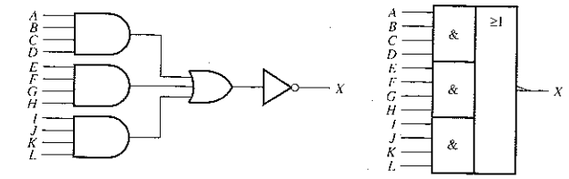
FIGURA P.17

CAPÍTULO 5

1. Véase la Figura P.18

3. **(a)** $X = ABB$ **(b)** $X = AB + B$ **(c)** $X = \overline{A} + B$
(d) $X = (A + B) + AB$ **(e)** $X = \overline{\overline{A}\,\overline{B}\,\overline{B}C}$ **(f)** $X = (A + B)(\overline{B} + C)$

FIGURA P.18



5. **(a)**

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

(b)

A	B	X
0	0	0
0	1	1
1	0	0
1	1	1

(c)

A	B	X
0	0	1
0	1	1
1	0	0
1	1	1

(d)

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

(e)

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

(f)

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

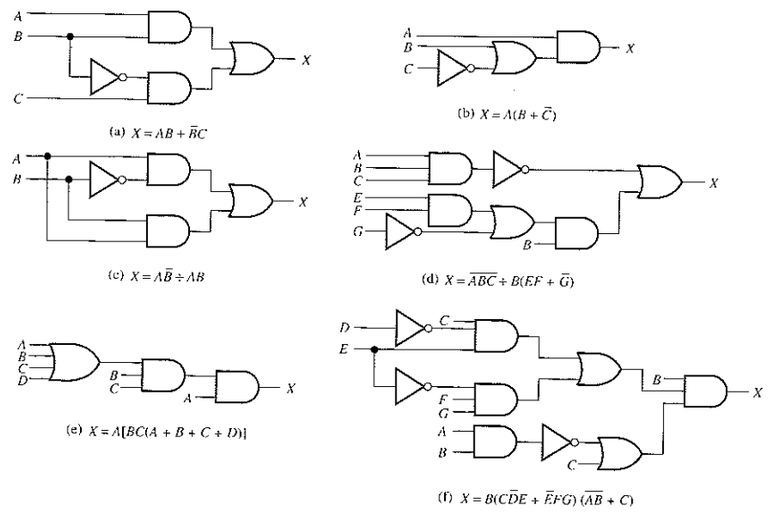
7. $X = \overline{A\overline{B} + \overline{A}B} = (\overline{A} + B)\,(A + \overline{B})$

9. Véase la Figura P.19

11. Véase la Figura P.20

13. $X = AB$

FIGURA P.19



(a) $X = AB + \overline{B}C$

(b) $X = A(B + \overline{C})$

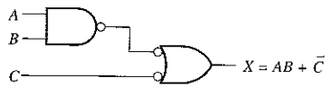
(c) $X = A\overline{B} + AB$

(d) $X = \overline{AB\overline{C}} + B(EF + \overline{G})$

(e) $X = A[BC(A + B + C + D)]$

(f) $X = B(C\overline{D}E + \overline{E}FG)\,(\overline{AB} + C)$

FIGURA P.20



15. **(a)** No se simplifica **(b)** No se simplifica
(c) $X = A$ **(d)** $X = \overline{A} + \overline{B} + \overline{C} + EF + \overline{G}$
(e) $X = ABC$ **(f)** $X = BC\overline{D}E + \overline{A}B\overline{E}FG + BC\overline{E}FG$

17. **(a)** $X = AC + AD + BC + BD$ **(b)** $X = \overline{A}CD + \overline{B}CD$
(c) $X = ABD + CD + E$ **(d)** $X = \overline{A} + B + D$
(e) $X = ABD + \overline{C}D + \overline{E}$ **(f)** $X = \overline{AC} + \overline{AD} + \overline{BC} + \overline{BD} + \overline{EG} + \overline{EH} + \overline{FG} + \overline{FH}$

19. Véase la Figura P.21

21. Véase la Figura P.22

23. Véase la Figura P.23

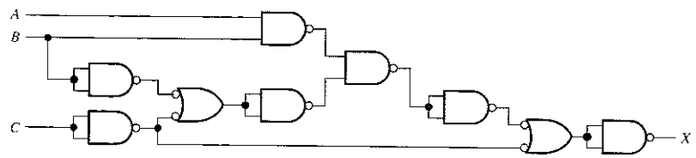


FIGURA P.21

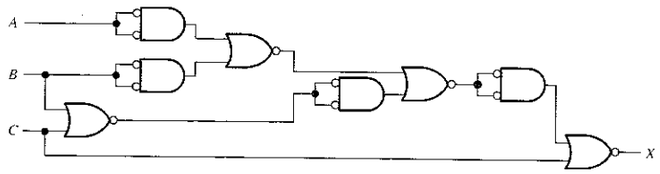
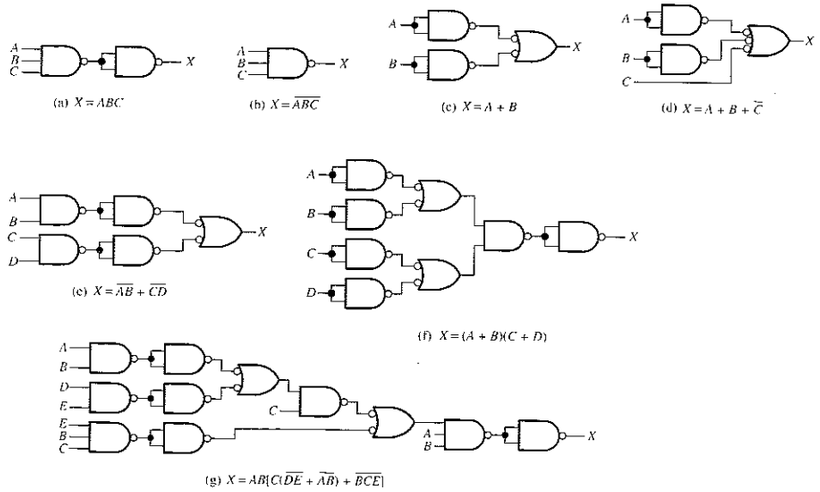


FIGURA P.22

FIGURA P.23



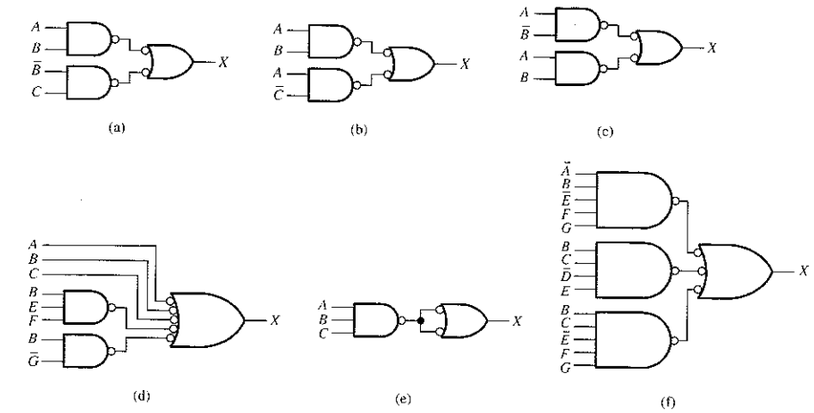


FIGURA P.24

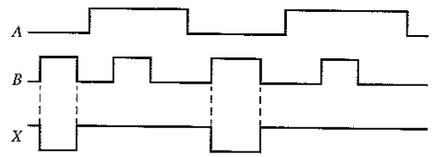


FIGURA P.25

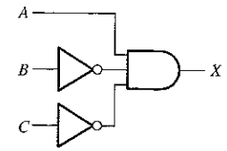


FIGURA P.26

25. Véase la Figura P.24

27. $X = A + \overline{B}$; véase la Figura P.25

29. $X = A\overline{B}\,\overline{C}$; véase la Figura P.26

31. La anchura del impulso de salida es mayor que el mínimo especificado.

33. $X = ABC + D\overline{E}$. Puesto que X es igual a la salida de G_3, G_1 o G_2 han fallado manteniendo su salida a nivel BAJO.

35. El pin 10 del 74AHC00 de la derecha está cortocircuitado a tierra.

37. **(a)** $X = E$; véase la Figura P.27 **(b)** $X = E$ **(c)** $X = E$

39. Véase la Figura P.28

41. Véase la Figura P.29, en la que se indica la lógica de M_4 modificada. La lógica de los restantes motores no varía.

43. X = lámpara encendida. A = interruptor frente a la puerta activado. B = interruptor detrás de la puerta activado. Véase la Figura P.30.

45. Véase la Figura P.31. Los inversores (no mostrados) se usan para convertir cada nivel ALTO de pulsación de tecla a nivel BAJO.

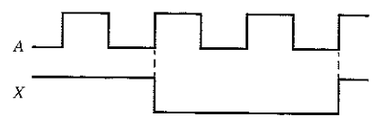


FIGURA P.27

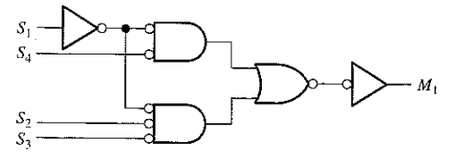


FIGURA P.28

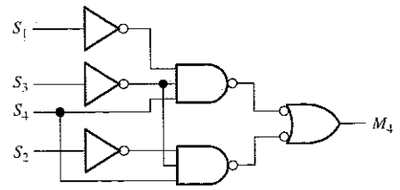


FIGURA P.29

FIGURA P.30

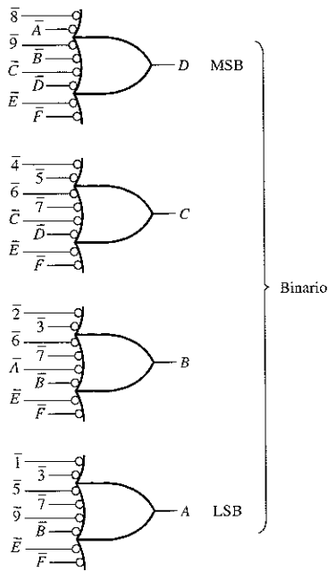


FIGURA P.31

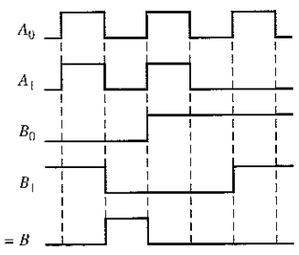


FIGURA P.32

CAPÍTULO 6

1. **(a)** $A \oplus B = 0, \Sigma = 1, (A \oplus B)C_{in} = 0, AB = 1, C_{out} = 1$
(b) $A \oplus B = 1, \Sigma = 0, (A \oplus B)C_{in} = 1, AB = 0, C_{out} = 1$
(c) $A \oplus B = 1, \Sigma = 1, (A \oplus B)(C_{in} = 0, AB = 0, C_{out} = 0$

3. **(a)** $\Sigma = 1, C_{out} = 0$; **(b)** $\Sigma = 1, C_{out} = 0$; **(c)** $\Sigma = 0, C_{out} = 1$; **(d)** $\Sigma = 1, C_{out} = 1$

5. 11100

7. $\Sigma_1 = 0110; \Sigma_2 = 1011; \Sigma_3 = 0110; \Sigma_4 = 0001; \Sigma_5 = 1010$

9. $A = B$ es un nivel ALTO cuando $A_0 = B_0$ y $A_1 = B_1$. Véase la Figura P.32

11. **(a)** $A > B = 1; A = B = 0; A < B = 0$ **(b)** $A < B = 1; A = B = 0; A > B = 0$
(c) $A = B = 1; A < B = 0; A > B = 0$

13. Véase la Figura P.33

15. $X = A_3A_2\overline{A}_1\overline{A}_0 + \overline{A}_3\overline{A}_2\overline{A}_1A_0 + A_3\overline{A}_2A_1$

17. Véase la Figura P.34

19. $A_3A_2A_1A_0 = 1011$. Código BCD no válido.

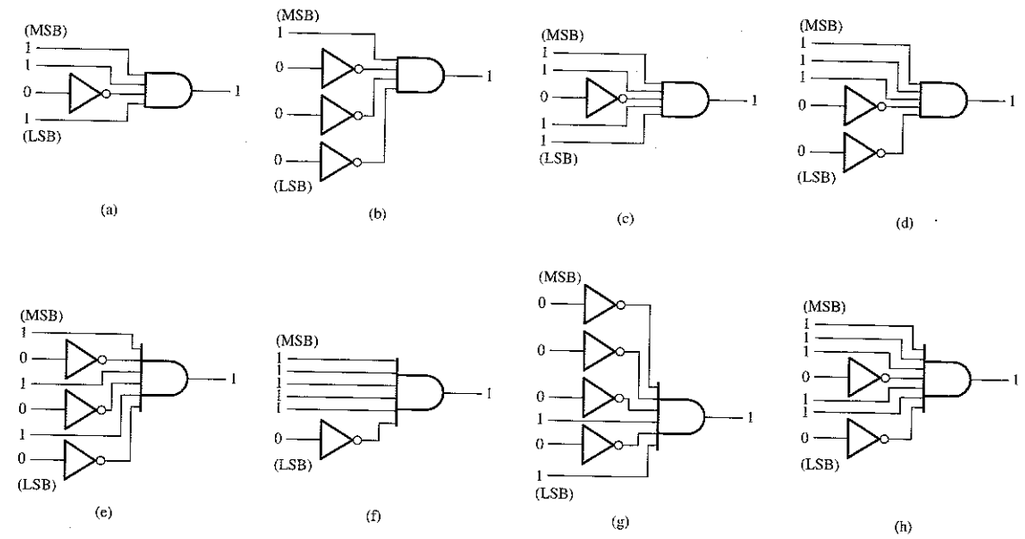
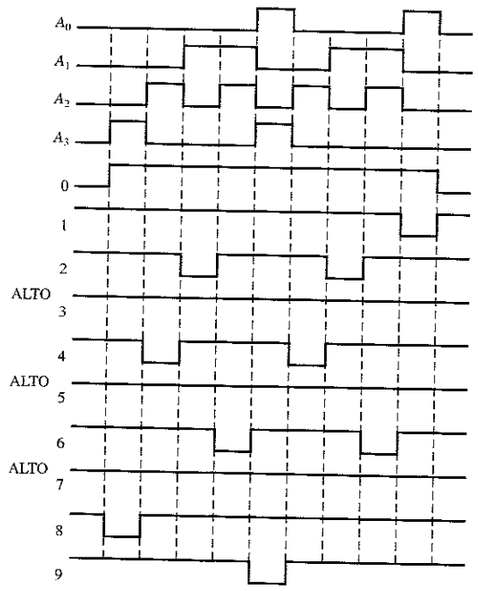


FIGURA P.33

FIGURA P.34



21. Véase la Figura P.35

23. **(a)** 1010000000 Gray
1100000000 binario
(b) 0011001100 Gray
0010001000 binario
(c) 1111000111 Gray
1010000101 binario
(d) 0000000001 Gray
0000000001 binario
Véase la Figura P.36

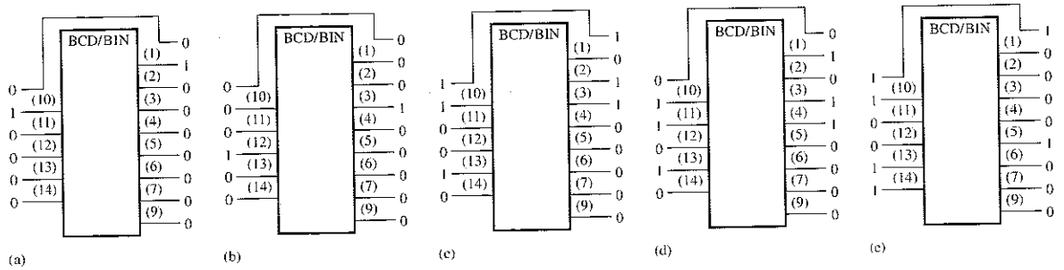


FIGURA P.35

25. Véase la Figura P.37

27. Véase la Figura P.38

29. Véase la Figura P.39

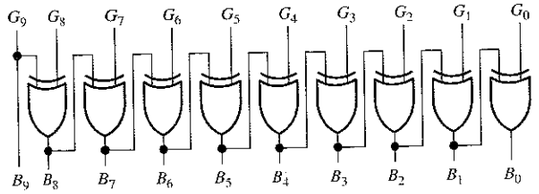


FIGURA P.36

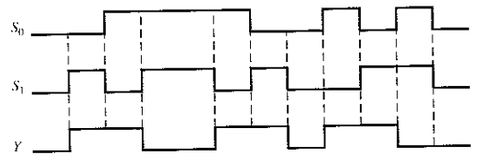


FIGURA P.37

FIGURA P.38

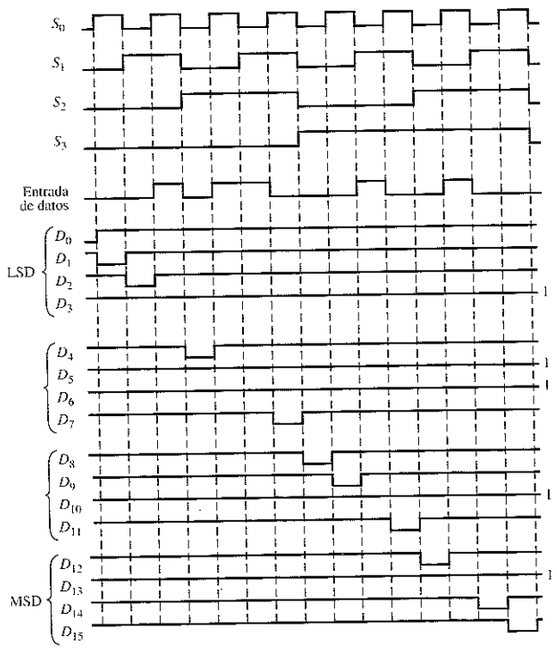
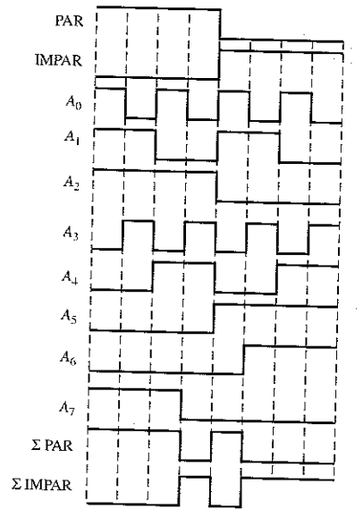


FIGURA P.39



31. **(a)** Correcto
(b) el segmento *g* está fundido; salida G abierta
(c) la salida del segmento *b* se mantiene a nivel BAJO.

33. **(a)** La entrada A_1 del sumador superior está en circuito abierto. Todos los valores binarios correspondientes al número BCD que tienen un valor de 0, 1, 4, 5, 8 o 9 se desplazan 2 unidades. El primer valor BCD para el que se detecta el error es 0000 0000.
(b) La salida de acarreo del sumador superior está en circuito abierto. Normalmente, no todos los valores que implican un acarreo de salida se desplazarán 32 unidades. El primer valor BCD para el que se detecta el error es 0000 0000.
(c) La salida Σ_4 del sumador superior está cortocircuitada a tierra. El mismo conjunto de valores anterior se reducirá en 16 unidades. El primer valor que indica esto es 0001 1000.
(d) La salida Σ_3 del sumador inferior está cortocircuitada a tierra: cada grupo de 16 valores, comenzando por 16, se reducirá en 16 unidades. El primer valor BCD que indica esto es 0001 0110.

TABLA P.7

S_2	S_1	S_0	Y	$\overline{Y}$
0	0	0	1	0
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	0	1

35. 1. Poner a nivel BAJO el pin 7 (habilitación).
2. Aplicar un nivel ALTO a D_0 y un nivel BAJO a D_1 hasta D_7.
3. Aplicar la secuencia binaria a las entradas de selección y comprobar Y e $\overline{Y}$ de acuerdo con la Tabla P.7.

4. Repetir la secuencia binaria de las entradas de selección para cada uno de los conjuntos de entradas de datos enumerados en la Tabla P.8. Un nivel ALTO en la salida Y sólo debería producirse para las combinaciones de las entradas de selección indicadas.

TABLA P.8

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	Y	$\overline{Y}$	S_2	S_1	S_0
L	H	L	L	L	L	L	L	1	0	0	0	1
L	L	H	L	L	L	L	L	1	0	0	1	0
L	L	L	H	L	L	L	L	1	0	0	1	1
L	L	L	L	H	L	L	L	1	0	1	0	0
L	L	L	L	L	H	L	L	1	0	1	0	1
L	L	L	L	L	L	H	L	1	0	1	1	0
L	L	L	L	L	L	L	H	1	0	1	1	1

37. Aplicar por turno un nivel ALTO a cada entrada de datos, D_0 hasta D_7, estando las demás entradas a nivel BAJO. Para cada nivel ALTO aplicado a una entrada de datos, seguir la secuencia de las ocho combinaciones binarias de las entradas de selección ($S_2S_1S_0$) y comprobar que se produce un nivel alto en la salida de datos correspondiente y que todas las restantes salidas de datos están a nivel BAJO.

39. Reemplazar cada 74LS00 por una puerta AND 74LS08.

41. $\Sigma = \overline{A}\,\overline{B}C_{in} + \overline{A}B\overline{C}_{in} + A\overline{B}\,\overline{C}_{in} + ABC_{in}$
$C_{out} = \overline{A}BC_{in} + A\overline{B}C_{in} + AB\overline{C}_{in} + ABC_{in}$
Véase la Figura P.40

43. Véase el diagrama de bloques de la Figura P.41

45. Véase la Figura P.42

47. Véase la Figura P.43

CAPÍTULO 7

1. **1.** Memoria programable de sólo lectura (PROM): matriz AND fija y matriz OR programable.
2. Matriz lógica programable (PLA): matrices AND y OR programables.
3. Lógica de matriz programable (PAL): matriz AND programable y matriz OR fija.
4. Matriz lógica genérica (GAL): matriz AND reprogramable y matriz OR fija.

3. Para $X_1 = \overline{A}BC$, fundido:
Fila 1, columnas 1, 3, 4, 5, 6
Fila 2, columnas 1, 2, 4, 5, 6
Fila 3, columnas 1, 2, 3, 4, 6
Para $X_2 = AB\overline{C}$, fundido:
Fila 4, columnas 2, 3, 4, 5, 6
Fila 5, columnas 1, 2, 4, 5, 6
Fila 6, columnas 1, 2, 3, 4, 5
Para $X_3 = \overline{A}B\overline{C}$, fundido:
Fila 7, columnas 1, 3, 4, 5, 6
Fila 8, columnas 1, 2, 4, 5, 6
Fila 9, columnas 1, 2, 3, 4, 5

5. Véase la Figura P.44

FIGURA P.40

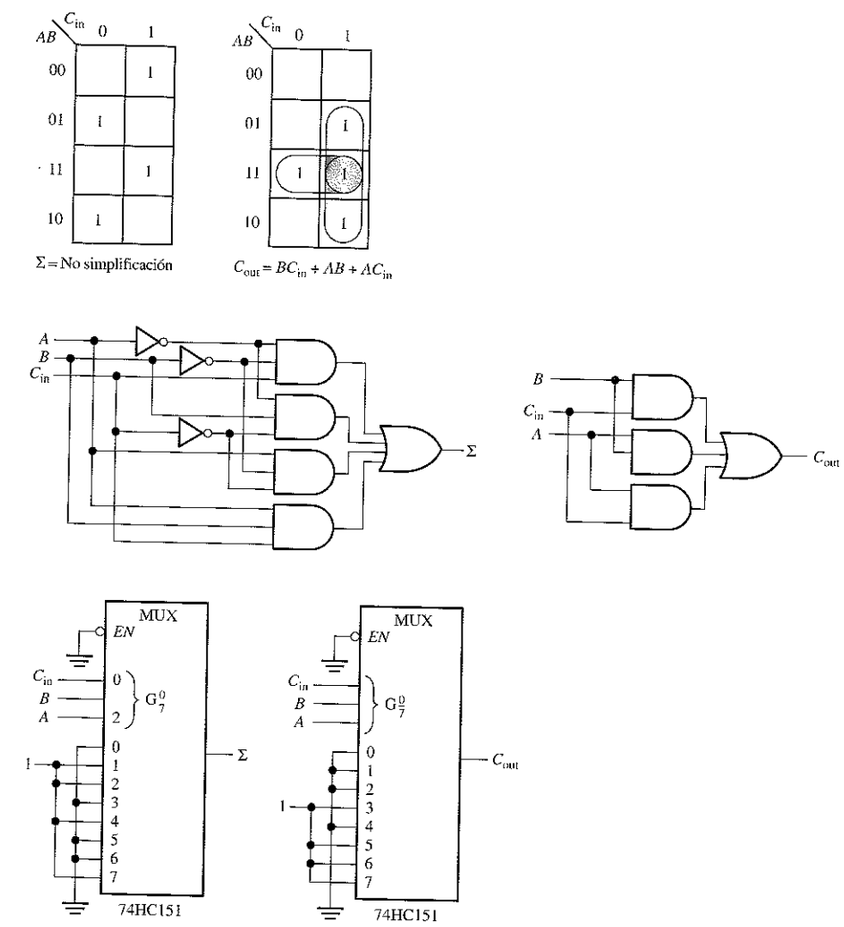


FIGURA P.41

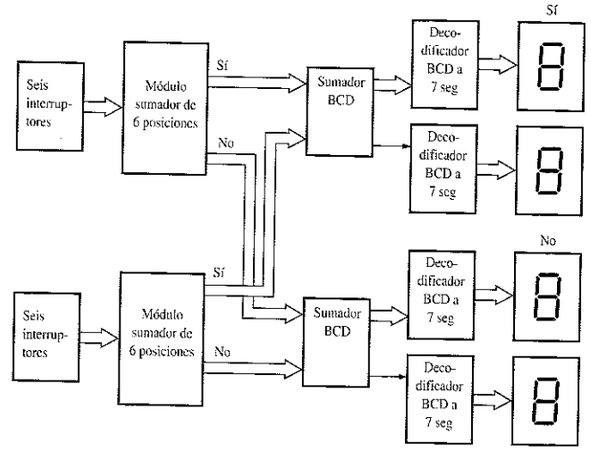


FIGURA P.42

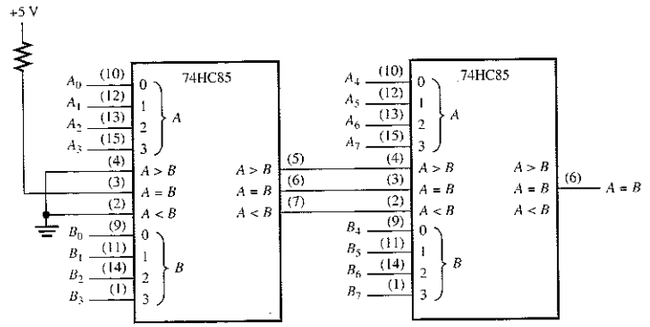


FIGURA P.43

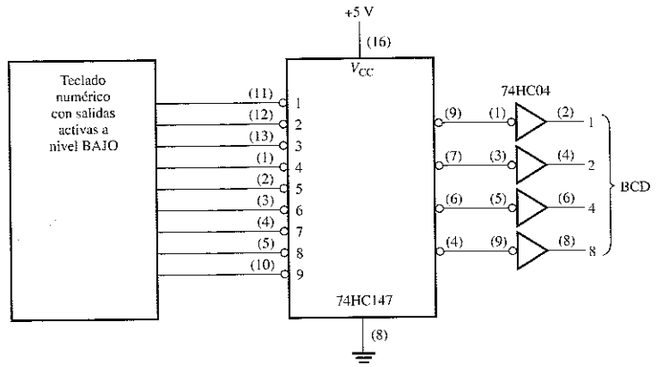
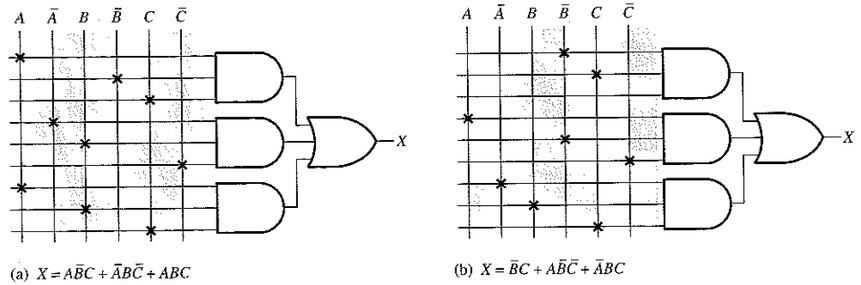


FIGURA P.44



(a) $X = A\bar{B}C + \bar{A}B\bar{C} + ABC$

(b) $X = \bar{B}C + A\bar{B}\bar{C} + \bar{A}BC$

7. La salida con polaridad programada en una PAL trabaja dejando intacto o abriendo selectivamente una celda de una entrada de una puerta XOR, haciendo que la puerta invierta o no las restantes entradas.

9. Véase la Figura P.45

11. Hay 7 salidas disponibles.

13. Las entradas de selección determinan el modo combinacional o secuencial, de salida o entrada, de salida invertida o no invertida.

15. **(a)** 8 **(b)** 8 **(c)** 16

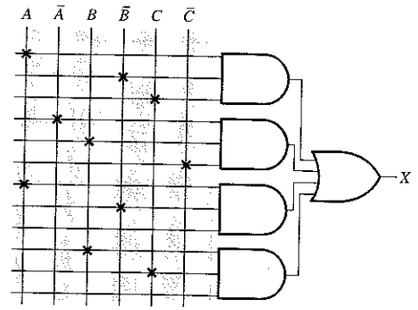
17. 8 términos producto, 15 variables

19. Computadora, programador, compilador

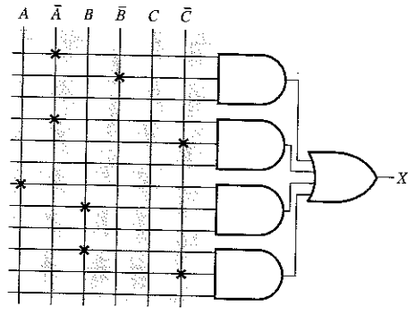
21. La ispGAL22V10 puede programarse dentro del circuito, mientras que la GAL22V10 debe insertarse en un programador.

23. **(a)** !A & !B & !C; **(b)** A # B # !C; **(c)** A & !(B # C);

25. **(a)** X = A & !B & C & !D & E # A & !B & C # !A & !B;
(b) Y = A & B & !C # D & E & F # !G & H & !I # J & !K & L;
(c) Z = (!A # !B # C # D) & (E # !F) # G;



(a) $X = A\bar{B}C + \bar{A}B\bar{C} + A\bar{B} + BC$



(b) $X = \bar{A}\bar{B} + \bar{A}\bar{C} + AB + B\bar{C}$

FIGURA P.45

27.
```
TRUTH_TABLE ([A, B] -> [X])
             [0, 0] -> [1];
             [0, 1] -> [0];
             [1, 0] -> [0];
             [1, 1] -> [1];
```

29.
```
Test_Vectors (BCD -> [a, b, c, d, e, f, g, h, i, j])
               0 -> [1, 0, 0, 0, 0, 0, 0, 0, 0, 0];
               1 -> [0, 1, 0, 0, 0, 0, 0, 0, 0, 0];
               2 -> [0, 0, 1, 0, 0, 0, 0, 0, 0, 0];
               3 -> [0, 0, 0, 1, 0, 0, 0, 0, 0, 0];
               4 -> [0, 0, 0, 0, 1, 0, 0, 0, 0, 0];
               5 -> [0, 0, 0, 0, 0, 1, 0, 0, 0, 0];
               6 -> [0, 0, 0, 0, 0, 0, 1, 0, 0, 0];
               7 -> [0, 0, 0, 0, 0, 0, 0, 1, 0, 0];
               8 -> [0, 0, 0, 0, 0, 0, 0, 0, 1, 0];
               9 -> [0, 0, 0, 0, 0, 0, 0, 0, 0, 1];
```

31. *Sí*, el diseño requiere sólo 2 entradas y 8 salidas.

33.
```
Module    Codificador_prioridad_decimal_BCD
Title     'Codificador_prioridad_decimal_BCD en una GAL16V8'

Codific_1          device          'P16V8'

I0, I1, I2, I3, I4                 pin 1,2,3,4,5;
I5, I6, I7, I8, I9                 pin 6,7,8,9,12;
Q0, Q1, Q2, Q3                     pin 16,17,18,19;

Equations
Q0 = I1 & !I2 & !I4 & !I6 & !I8 # I3 & !I4 & !I6 & !I8 # I5 & !I6 &
     !I8 # I7 & !I8 # I9;
Q1 = I2 & !I4 & !I5 & !I8 & !I9 # I3 & !I4 & !I5 & !I8 & !I9 #
     I6 & !I8 & !I9 # I7 & !I8 & !I9;
```

```
Q2 = I4 & !I8 & !I9 # I5 & !I8 & !I9 # I6 & !I8 & !I9 # I7 & !I8 & !I9;
Q3 = I8 # I9;
```

CAPÍTULO 8

1. Véase la Figura P.46

3. Véase la Figura P.47

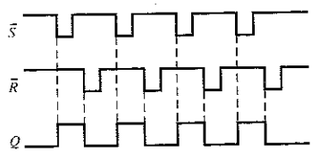


FIGURA P.46

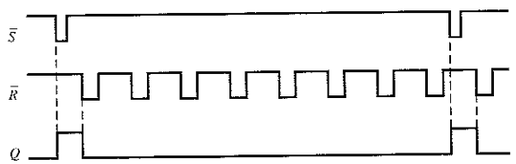


FIGURA P.47

5. Véase la Figura P.48

7. Véase la Figura P.49

9. Véase la Figura P.50

11. Véase la Figura P.51

13. Véase la Figura P.52

15. Véase la Figura P.53

17. Véase la Figura P.54

19. Véase la Figura P.55

21. Véase la Figura P.56

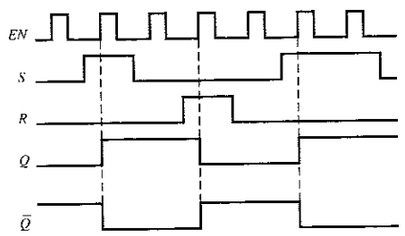


FIGURA P.48

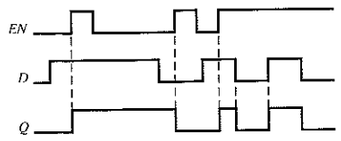


FIGURA P.49

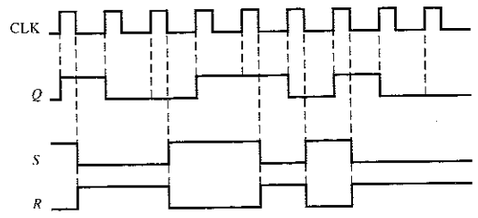


FIGURA P.50

FIGURA P.51

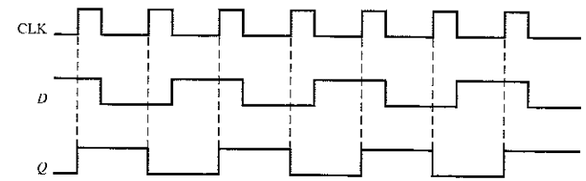


FIGURA P.52

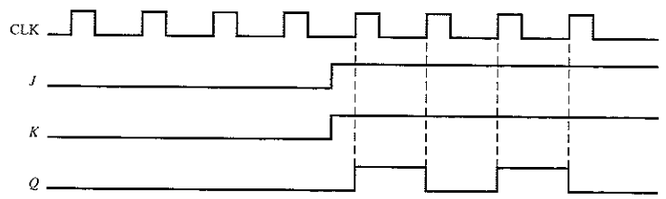
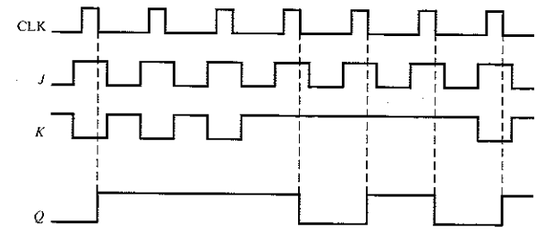


FIGURA P.53



23. Véase la Figura P.57

25. 14,9 MHz

27. 150 mA, 750 mW

29. Divisor por 2. Véase la Figura P.58

31. 4,62 µs

33. $C_1 = 1$ µF, $R_1 = 227$ kΩ (utilice 220 kΩ). Véase la Figura P.59

35. $R_1 = 18$ kΩ. $R_2 = 9{,}1$ kΩ

37. La conexión del pin 6 al pin 10 y la conexión de tierra están invertidos en el prototipo de la tarjeta.

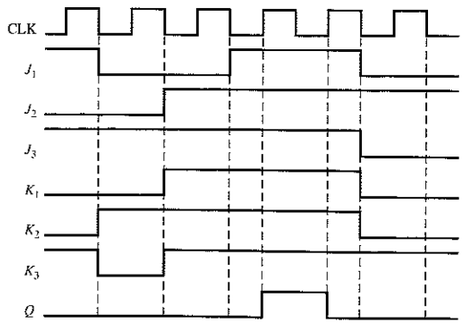


FIGURA P.54

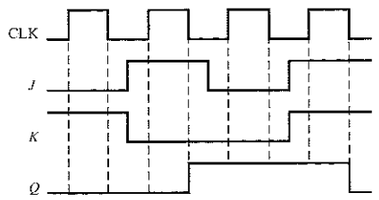


FIGURA P.55

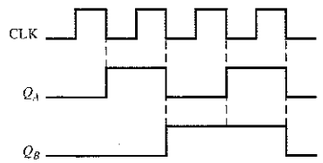


FIGURA P.56

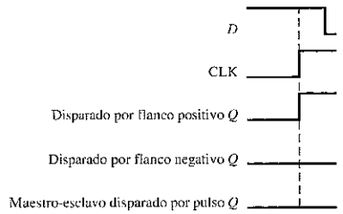


FIGURA P.57

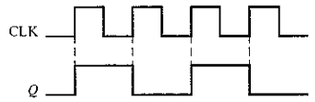


FIGURA P.58

FIGURA P.59

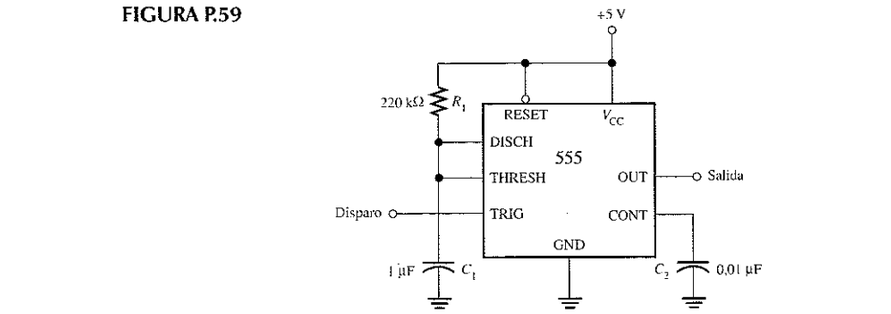


FIGURA P.60

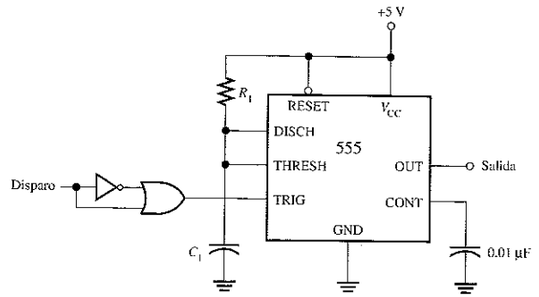


FIGURA P.61

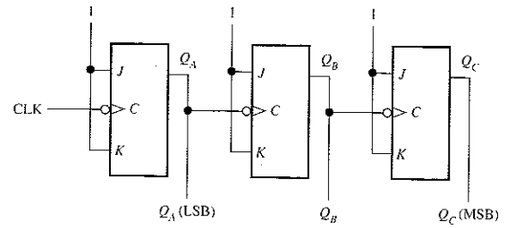


FIGURA P.62

39. $\overline{CLR}$ está cortocircuitada(o) a tierra.

41. Véase la Figura P.60. Los retardos no se indican.

43. Véase la Figura P.61
4 s: $C_1 = 1\ \mu F$, $R_1 = 3{,}63\ M\Omega$ (utilizar 3,9 MΩ)
25 s: $C_1 = 2{,}2\ \mu F$, $R_1 = 10{,}3 M\Omega$ (utilizar 10 MΩ)

45. Véase la Figura P.62

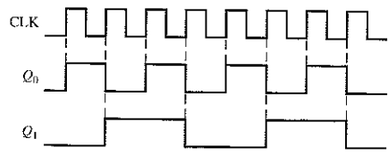
47. Los cambios son: (1) incluir un temporizador de 15 segundos; (2) decodificación para disparar el temporizador de 15 segundos, y (3) un decodificador para incorporar el estado de la señal de encendido.

CAPÍTULO 9

1. Véase la Figura P.63

3. El retardo en el caso peor es 24 ns. Se produce cuando todos los flip-flops cambian del estado 011 al 100, o del 111 al 000.

FIGURA P.63



5. 8 ns.

7. Inicialmente cada flip-flop está en estado de reset.

En CLK1: $J_0 = K_0 = 1$, por tanto Q_0 se pone a 1
$J_1 = K_1 = 0$, por tanto Q_1 se mantiene a 0
$J_2 = K_2 = 0$, por tanto Q_2 se mantiene a 0
$J_3 = K_3 = 0$, por tanto Q_3 se mantiene a 0

En CLK2: $J_0 = K_0 = 1$, por tanto Q_0 se pone a 0
$J_1 = K_1 = 1$, por tanto Q_1 se pone a 1
$J_2 = K_2 = 0$, por tanto Q_2 se mantiene a 0
$J_3 = K_3 = 0$, por tanto Q_3 se mantiene a 0

En CLK3: $J_0 = K_0 = 1$, por tanto Q_0 se pone a 1
$J_1 = K_1 = 0$, por tanto Q_1 se mantiene a 1

$J_2 = K_2 = 0$, por tanto Q_2 se mantiene a 0

$J_3 = K_3 = 0$, por tanto Q_3 se mantiene a 0

Siguiendo este proceso para los siete impulsos de reloj se verá que el contador progresa a través de una secuencia BCD.

9. Véase la Figura P.64

11. Véase la Figura P.65

13. Véase la Figura P.66

15. La secuencia es 0000, 1111, 1110, 1101, 1010, 0101. El contador "se bloquea" en los estados 1010 y 0101 y alterna entre ellos.

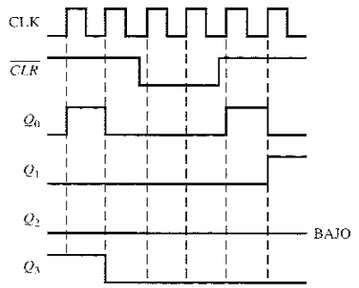


FIGURA P.64

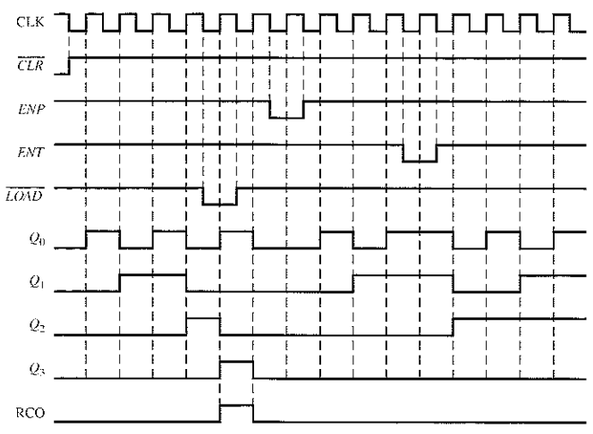
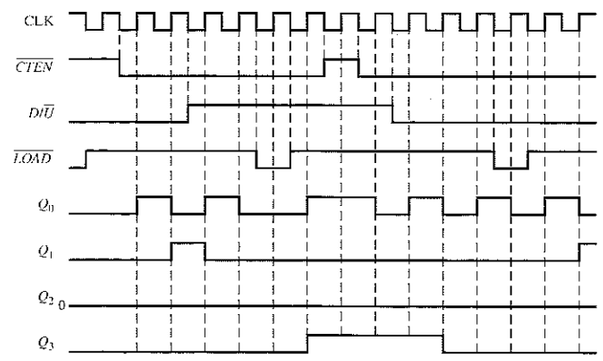


FIGURA P.65

FIGURA P.66



17. Véase la Figura P.67

19. Véase la Figura P.68

21. Véase la Figura P.69 para la división por 10.000. Añadir un contador DIV10 más para crear un divisor por 100.000.

23. Véase la Figura P.70

25. CLK2, salida 0; CLK4, salidas 2, 0; CLK6, salida 4; CLK8, salidas 6, 4, 0; CLK10, salida 8, CLK12, salidas 10, 8; CLK14, salida 12; CLK16, salidas 14, 12, 8.

27. Se produce un glitch en la salida de la puerta AND en la transición de 111 a 000. Se elimina aplicando la operación AND a $\overline{CLK}$ con las salidas del contador (validación, strobe) o utilizando el código Gray.

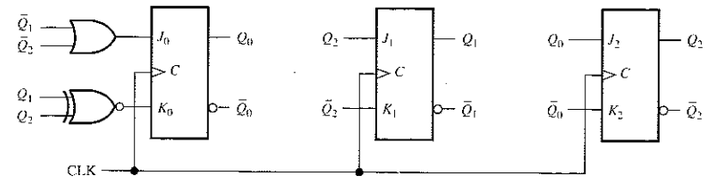


FIGURA P.67

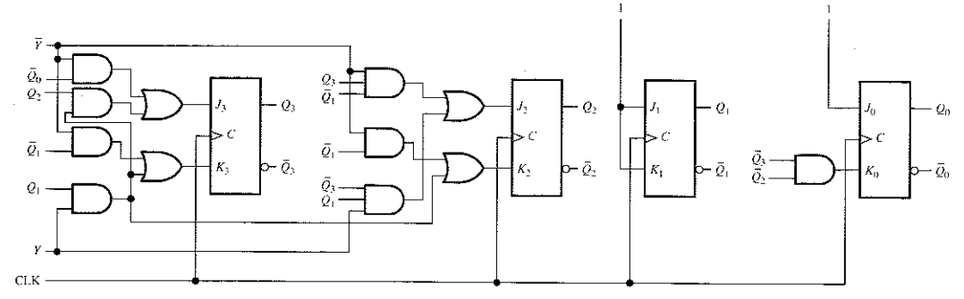
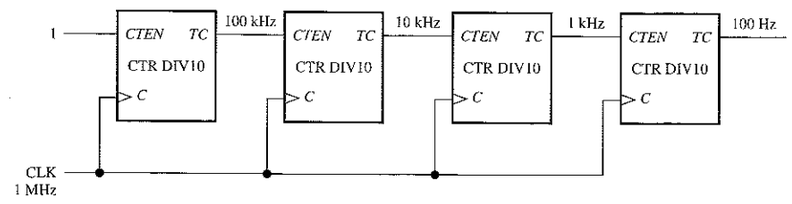


FIGURA P.68

FIGURA P.69



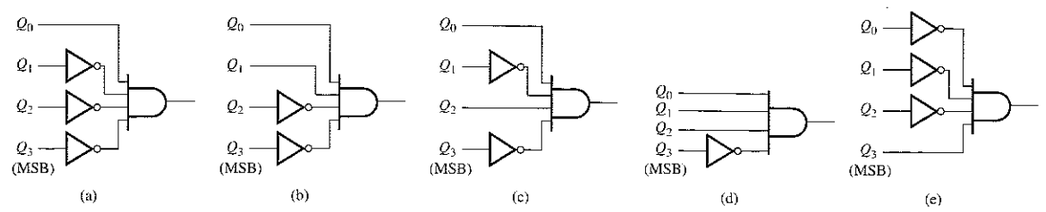


FIGURA P.70

29. Decenas de horas: 0001
Unidades de horas: 0010
Decenas de minutos: 0000

Unidades de minutos: 0001
Decenas de segundos: 0000
Unidades de segundos: 0010

31. 64

33. **(a)** Q_0 y Q_1 no cambian desde su estado inicial.
(b) El funcionamiento es normal excepto para el estado flotante de Q_0.
(c) La forma de onda de Q_0 es normal y Q_1 permanece en el estado inicial.
(d) Funcionamiento normal.
(e) El contador no cambiará desde su estado inicial.

35. La entrada K de FF1 debe conectarse a tierra en lugar de la entrada J. Comprobar un error de conexión.

37. La entrada Q_0 y la puerta AND están en circuito abierto y actúan como un nivel ALTO.

39. Véase la Tabla P.9

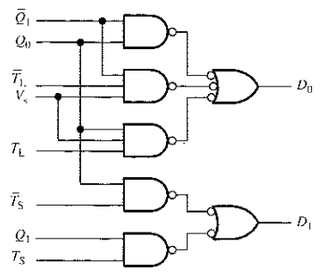
TABLA P.9

Etapa	Abierto	Cuenta cargada	f_{salida}
1	0	63C1	250,006 Hz
1	1	63C2	250,012 Hz
1	2	63C4	250,025 Hz
1	3	63C8	250,050 Hz
2	0	63D0	250,100 Hz
2	1	63E0	250,200 Hz
2	2	63C0	250 Hz
2	3	63C0	250 Hz
3	0	63C0	250 Hz
3	1	63C0	250 Hz
3	2	67C0	256,568 Hz
3	3	6BC0	263,491 Hz
4	0	73C0	278,520 Hz
4	1	63C0	250 Hz
4	2	63C0	250 Hz
4	3	E3C0	1,383 kHz

41. La puerta 6 de decodificación interpreta la cuenta 4 como 6 (0110) y borra el contador poniéndolo a 0 (realmente 0010, puesto que Q_1 está en circuito abierto). La secuencia aparente de la parte de las unidades del contador es 0010, 0011, 0010, 0011, 0110.

43. Véase la Figura P.71

FIGURA P.71



45. Aumentar 2,4 veces la constante de tiempo $R_{EXT}C_{EXT}$ del monoestable de 25 segundos.

47. Véase la Figura P.72

49. Véase la Figura P.73

51. Véase la Figura P.74

53. Véase la Figura P.75

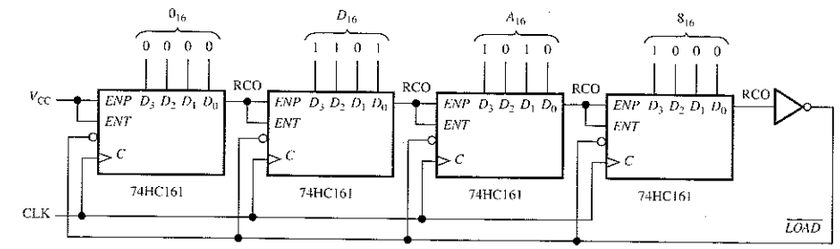


FIGURA P.72

FIGURA P.73

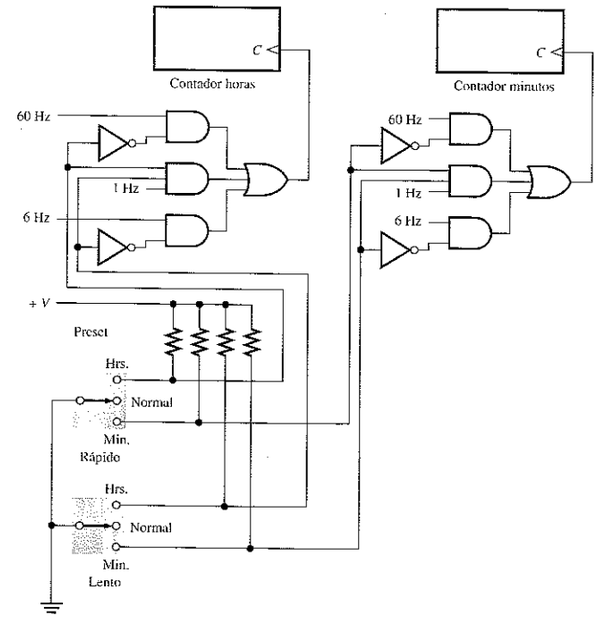


FIGURA P.74

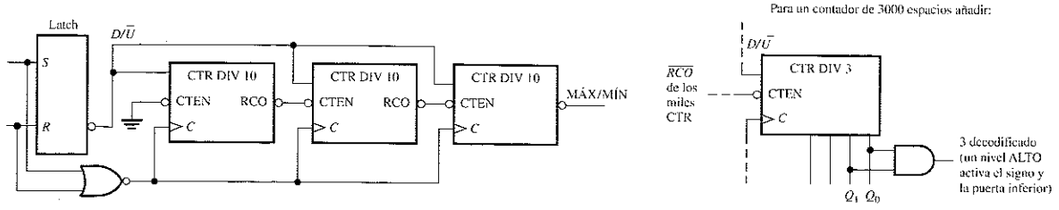


FIGURA P.75

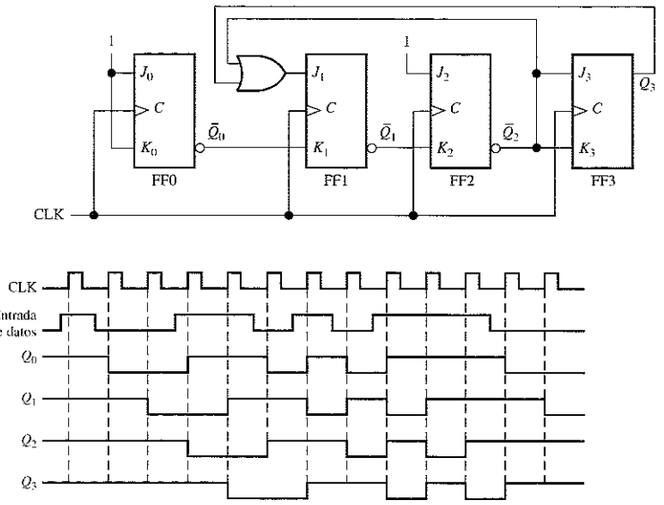


FIGURA P.76

CAPÍTULO 10

1. Los registros de desplazamiento almacenan datos binarios.

3. Véase la Figura P.76

5. Inicialmente: 101001111000

CLK1:	010100111100
CLK2:	001010011110
CLK3:	000101001111
CLK4:	000010100111
CLK5:	100001010011
CLK6:	110000101001
CLK7:	111000010100
CLK8:	011100001010
CLK9:	001110000101
CLK10:	000111000010
CLK11:	100011100001
CLK12:	110001110000

7. Véase la Figura P.77

9. Véase la Figura P.78

11. Véase la Figura P.79

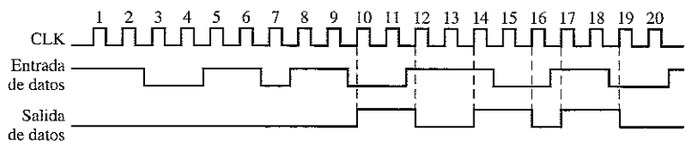


FIGURA P.77

FIGURA P.78

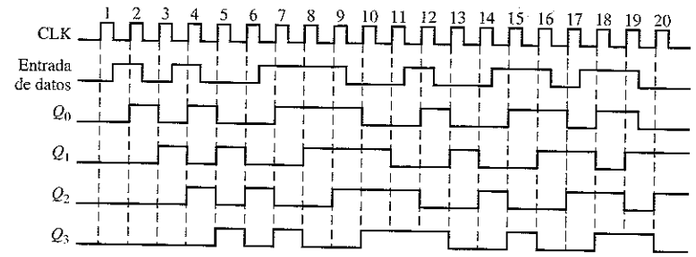
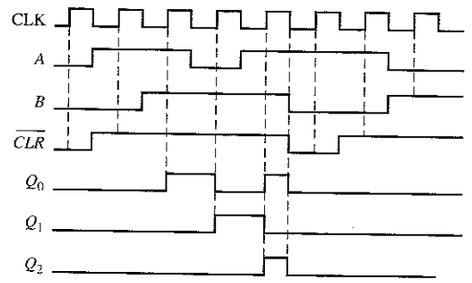


FIGURA P.79



Q_3 hasta Q_7 permanecen a nivel BAJO

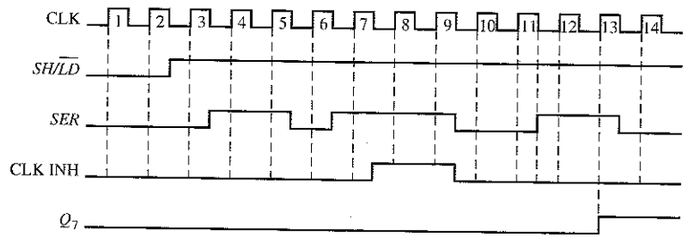
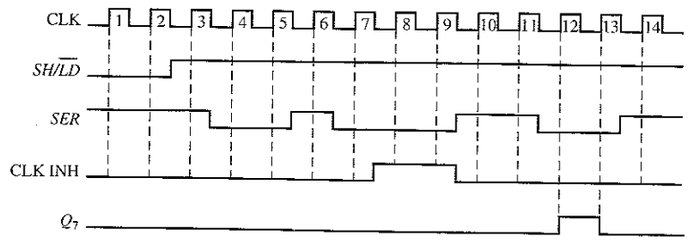


FIGURA P.80

FIGURA P.81



13. Véase la Figura P.80

15. Véase la Figura P.81

17. Véase la Figura P.82

19. Inicialmente (76):01001100

CLK1:	10011000	izquierda
CLK2:	01001100	derecha
CLK3:	00100110	derecha
CLK4:	00010011	derecha
CLK5:	00100110	izquierda
CLK6:	01001100	izquierda
CLK7:	00100110	derecha
CLK8:	01001100	izquierda
CLK9:	00100110	derecha
CLK10:	01001100	izquierda
CLK11:	10011000	izquierda

21. Véase la Figura P.83

23. **(a)** 3 **(b)** 5 **(c)** 7 **(d)** 8

25. Véase la Figura P.84

27. Véase la Figura P.85

29. Se puede producir un código incorrecto.

31. La entrada D_3 está en circuito abierto.

33. **(a)** No se produce un impulso de reloj cuando se cierra el interruptor porque falla la puerta NAND (negativa-OR) o el monoestable. La entrada de reloj (C) del registro del código tecleado está en circuito abierto. La entrada $SH/\overline{LD}$ del registro del código tecleado está en circuito abierto.

(b) El diodo de la tercera fila está en circuito abierto. La salida Q_2 del contador en anillo está en circuito abierto.

(c) La entrada de la puerta NAND (negativa-OR) conectada a la primera columna está en circuito abierto o cortocircuitada.

(d) La entrada "2" del decodificador de columnas está en circuito abierto.

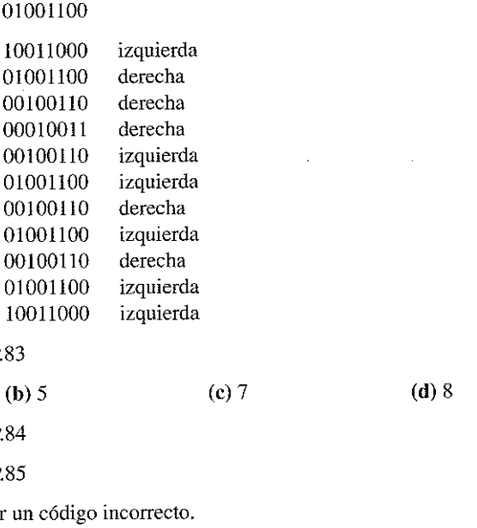


FIGURA P.82

FIGURA P.83

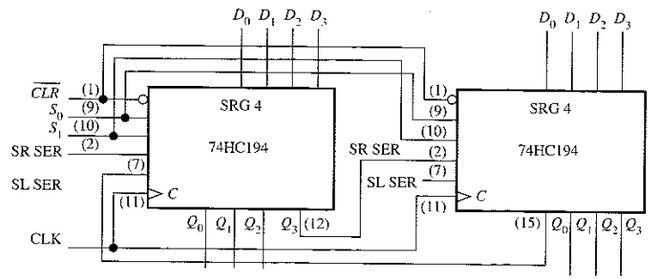
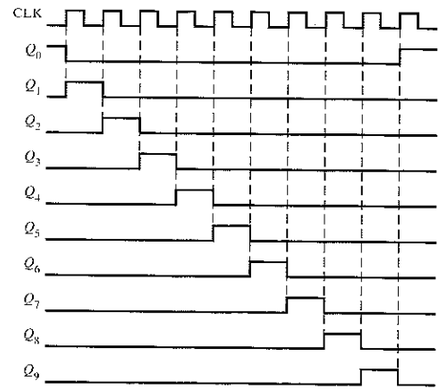


FIGURA P.84



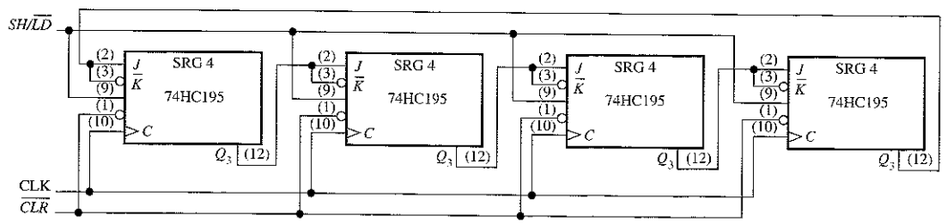


FIGURA P.85

35. **(a)** El contenido del registro de salida permanece constante.

(b) El contenido de ambos registros no cambia.

(c) La salida de la tercera etapa del registro de salida de datos permanece a nivel alto.

(d) El generador de reloj se desactiva después de cada impulso por lo que el flip-flop pasa continuamente del estado SET al estado RESET.

37. Registro de desplazamiento A: 1001
Registro de desplazamiento B: 11100

39. La puerta G_1 tiene una salida en circuito abierto o cortocircuitada, o una entrada en circuito abierto. El monoestable A falla

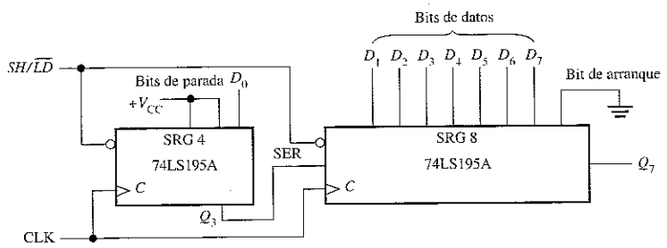
41. Flip-flop de control: 7476
Generador de reloj: 555
Contador: 74LS163

Registro de entrada de datos: 74LS164
Registro de salida de datos: 74LS199
Monoestable: 74121

43. Véase la Figura P.86

45. Véase la Figura P.87

FIGURA P.86



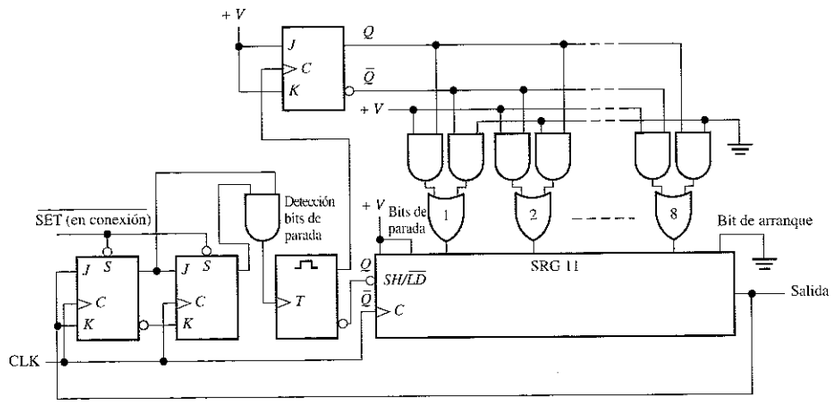


FIGURA P.87

CAPÍTULO 11

1. Salidas combinacionales activas a nivel BAJO; salidas combinacionales activas a nivel ALTO; salidas secuenciales activas a nivel BAJO y salidas secuenciales activas a nivel ALTO.

3. Multiplexor

5. **(a)** El pin 15 es una salida combinacional asociada a la variable X.
(b) El pin 20 es una salida secuencial asociada con la variable Q_2.
(c) El pin 18 es una salida combinacional activa a nivel BAJO asociada con la variable A.
(d) El pin 21 es una salida secuencial activa a nivel ALTO asociada con la variable Q_1.

7. Q1.CLK = Reloj;

9.
```
Test_vectors ([Clock,Clear,Din]->[ Q0, Q1, Q2, Q3, Q4, Q5, Q6,Qout])
                [ .x. ,  0  ,.x.]->[ 0 , 0 , 0 , 0 , 0 , 0 , 0 ,  0 ];
                [ .c. ,  1  ,.1.]->[ 1 , 0 , 0 , 0 , 0 , 0 , 0 ,  0 ];
                [ .c. ,  1  ,.0.]->[ 0 , 1 , 0 , 0 , 0 , 0 , 0 ,  0 ];
                [ .c. ,  1  ,.1.]->[ 1 , 0 , 1 , 0 , 0 , 0 , 0 ,  0 ];
                [ .c. ,  1  ,.0.]->[ 0 , 1 , 0 , 1 , 0 , 0 , 0 ,  0 ];
```



```
                [ .c. ,  1  ,.1.]->[ 1 , 0 , 1 , 0 , 1 , 0 , 0 ,  0 ];
                [ .c. ,  1  ,.0.]->[ 0 , 1 , 0 , 1 , 0 , 1 , 0 ,  0 ];
                [ .c. ,  1  ,.1.]->[ 1 , 0 , 1 , 0 , 1 , 0 , 1 ,  0 ];
                [ .c. ,  1  ,.0.]->[ 0 , 1 , 0 , 1 , 0 , 1 , 0 ,  1 ];
```

11. Los vectores de prueba son idénticos a los de la GAL22V10 del problema 9.

13. El registro de 8 bits del problema 12 no se puede implementar en una GAL16V8 porque los pines de entrada son insuficientes (se requieren 9 y la GAL16V8 tiene 8).

15. Contador de 8 bits.

17. Para un único contador ascendente en código Gray las ecuaciones lógicas son (utilizando la notación ABEL):

Q0 = !Q2 & !Q1 & !Q0 # !Q2 & !Q1 & Q0 # Q2 & Q1 & !Q0 # Q2 & Q1 & Q0
   = > !Q2 & !Q1 # Q2 & Q1
Q1 = !Q2 & !Q1 & Q0 # !Q2 & Q1 & Q0 # !Q2 & Q1 & !Q0 # Q2 & Q1 & !Q0
   => !Q2 & Q0 # Q1 & !Q0
Q2 = !Q2 & Q1 & !Q0 # Q2 & Q1 & !Q0 # Q2 & Q1 & Q0 # Q2 & !Q1 & Q0
   => Q1 & !Q0 # Q2 & Q0

El fichero de entrada en ABEL para un circuito GAL22V10 es entonces

```
Module        Contador_ascd_cód_Gray_3_bit
Title         'Contador_ascd_cód_Gray_3_bit en GAL22V10'
   "Declaración de dispositivo
        Contador  DEVICE    'P22V10'
"Declaración de pines
   Clock,Clear          Pin 1,2;
   Q0,Q1,Q2             Pin 21,22,23 ISTYPE 'reg,buffer';
Equations
   [Q0,Q1,Q2].CLK = Clock;
   [Q0,Q1,Q2].AR = !Clear;
   Q0 := !Q2.FB&!Q1.FB # Q2.FB&Q1.FB;
   Q1 := !Q2.FB&Q0.FB # Q1.FB&!Q0.FB;
   Q2 := Q1.FB&Q0.FB # Q2.FB&Q0.FB;
Test_vectors ([Clock ,  Clear]->[Q2 , Q1 , Q0 ])
                [ .x. ,    0   ]->[ 0 ,  0 ,  0 ];
                [ .c. ,    1   ]->[ 0 ,  0 ,  1 ];
                [ .c. ,    1   ]->[ 0 ,  1 ,  1 ];
                [ .c. ,    1   ]->[ 0 ,  1 ,  0 ];
                [ .c. ,    1   ]->[ 1 ,  1 ,  0 ];
                [ .c. ,    1   ]->[ 1 ,  1 ,  1 ];
                [ .c. ,    1   ]->[ 1 ,  0 ,  1 ];
                [ .c. ,    1   ]->[ 1 ,  0 ,  0 ];
                [ .c. ,    1   ]->[ 0 ,  0 ,  0 ];
END
```

19.
```
Module        Contador_ascd_cód_Gray_3_bit
Title         'Contador_ascd_cód_Gray_3_bit en GAL22V10'
"Declaración de dispositivo
   Contador      DEVICE       'P22V10'
"Declaración de pines
   Clock,Clear                Pin 1,2;
   Q0,Q1,Q2                   Pin 21,22,23 ISTYPE 'reg,buffer';
```

```
    COUNT  = [Q2,Q1,Q0];
    A      = ^B000;
    B      = ^B001;
    C      = ^B011;
    D      = ^B010;
    E      = ^B110;
    F      = ^B111;
    G      = ^B101;
    H      = ^B100;
Equations
    COUNT.CLK = Clock;
    COUNT.AR = !Clear;

State_diagram COUNT
    State A : GOTO B;
    State B : GOTO C;
    State C : GOTO D;
    State D : GOTO E;
    State E : GOTO F;
    State F : GOTO G;
    State G : GOTO H;
    State H : GOTO A;
Test_vectors ([Clock, Clear]->[Q2, Q1, Q0 ])
              [ .x. ,   0   ]->[ 0, 0 , 0  ];
              [ .c. ,   1   ]->[ 0, 0 , 1  ];
              [ .c. ,   1   ]->[ 0, 1 , 1  ];
              [ .c. ,   1   ]->[ 0, 1 , 0  ];
              [ .c. ,   1   ]->[ 1, 1 , 0  ];
              [ .c. ,   1   ]->[ 1, 1 , 1  ];
              [ .c. ,   1   ]->[ 1, 0 , 1  ];
              [ .c. ,   1   ]->[ 1, 0 , 0  ];
              [ .c. ,   1   ]->[ 0, 0 , 0  ];
END
```

21.

```
Module        Contador_ascd_cód_Gray_3_bit

Title         'Contador_ascd_cód_Gray_3_bit en GAL16V8'
"Declaración de dispositivo
   Contador       DEVICE       'P16V8'
"Declaración de pines
       Clock,Clear             Pin 1,11;
       Q0,Q1,Q2                Pin 17,18,19 ISTYPE 'reg,buffer';
Equations
       [Q0,Q1,Q2].CLK = Clock;
       [Q0,Q1,Q2].AR  = !Clear;

Truth_table ([Clock,  Q2, Q1,  Q0]:->[Q2 , Q1 , Q0 ])
             [ .c. ,   0,  0,  0 ]:->[ 0 ,  0 ,  1  ];
             [ .c. ,   0,  0,  1 ]:->[ 0 ,  1 ,  1  ];
             [ .c. ,   0,  1,  1 ]:->[ 0 ,  1 ,  0  ];
             [ .c. ,   0,  1,  0 ]:->[ 1 ,  1 ,  0  ];
             [ .c. ,   1,  1,  0 ]:->[ 1 ,  1 ,  1  ];
             [ .c. ,   1,  1,  1 ]:->[ 1 ,  0 ,  1  ];
             [ .c. ,   1,  0,  1 ]:->[ 1 ,  0 ,  0  ];
             [ .c. ,   1,  0,  0 ]:->[ 0 ,  0 ,  0  ];

Test_vectors ([Clock ,  Clear  ]:->[ Q2 , Q1 ,  Q0 ])
              [ .x.  ,    0    ]:->[ 0 ,  0 ,  0  ];
```



```
              [ .c.  ,    1    ]:->[ 0 ,  0 ,  1  ];
              [ .c.  ,    1    ]:->[ 0 ,  1 ,  1  ];
              [ .c.  ,    1    ]:->[ 0 ,  1 ,  0  ];
              [ .c.  ,    1    ]:->[ 1 ,  1 ,  0  ];
              [ .c.  ,    1    ]:->[ 1 ,  1 ,  1  ];
              [ .c.  ,    1    ]:->[ 1 ,  0 ,  1  ];
              [ .c.  ,    1    ]:->[ 1 ,  0 ,  0  ];
              [ .c.  ,    1    ]:->[ 0 ,  0 ,  0  ];
END
```

23.
(a) Estado CERRADO1: el sistema pasa al estado ESPERA1.
(b) Estado ESPERA1: el sistema permanece en el estado ESPERA1.
(c) Estado SUBIDA: el sistema permanece en el estado SUBIDA hasta que se genera la señal LLEGADA.
(d) Estado ESPERA2: el sistema permanece en el estado ESPERA2.
(e) Estado CERRADO2: el sistema pasa al estado ESPERA2.
(f) Estado BAJADA: el sistema permanece en el estado BAJADA hasta que se genera la señal de LLEGADA.

25.

```
Module        Control_estados_ascensor

Title         'Lógica de control para el sistema del ascensor para dos
              pisos'
"Declaración de dispositivo
   Contr_ascensor      DEVICE         'P22V10'
"Declaración de pines
   CLK, !OE                          pin 1,13;
   LLAMAR1,LLAMAR2                   pin 2,3;
   PISO2,PISO1, ABRIR, LLEGADA       pin 4,5,6,7;
   L1PB,L1PB_                        pin 20,21 ISTYPE 'com,buffer';
   L2PB,L2PB_                        pin 22,23 ISTYPE 'com,buffer';
   PUERTA,MOVIM,DIR                  pin 14,15,16 ISTYPE 'reg,buffer';

   "Definición de estados

      ESTADCON  = [PUERTA,MOVIM,DIR];
      ESPERA1   = ^B000;
      ABIERTO1  = ^B100;
      SUBIDA    = ^B110;
      ESPERA2   = ^B001;
      ABIERTO2  = ^B101;
      BAJADA    = ^B111;

   Equations
      ESTADCON.CLK = CLK;
      ESTADCON.AR  = OE;
      L1PB      = LLAMAR1 # !DIR.FB&ABRIR # DIR.FB&PISO1 # !L1PB_;
      L1PB_     = !PUERTA.FB&!MOVIM.FB&!DIR.FB # !L1PB;
      L2PB      = LLAMAR2 # DIR.FB&ABRIR # !DIR.FB&PISO2 # !L2PB_;
      L2PB_     = !PUERTA.FB&!MOVIM.FB&DIR.FB # !L2PB;

State_diagram ESTADCON
 State ESPERA1 : if (L2PB) then SUBIR else CERRADO1;
 State CERRADO1: if (L2PB) then SUBIR else if L1PB then ESPERA1 else CERRADO1;
 State SUBIDA  : if (LLEGADA) then ESPERA2 else SUBIR;
 State ESPERA2 : if (L1PB) then BAJAR else CERRADO2;
 State CERRADO2: if (L1PB) then BAJAR else if (L1PB) then ESPERA2 else
                 CERRADO2;
 State BAJADA  : if (LLEGADA) then ESPERA1 else BAJAR;
```

```
Test_vectors
   ([CLK,LLAMAR1,LLAMAR2,PISO1,PISO 2,ABRIR,LLEGADA]->[PUERTA,MOVIM,DIR])
    [.c.,  0  ,  0  ,  0  ,  0  ,  0  ,  1  ]->[ .x. ,  .x. ,.x.];
    [.c.,  1  ,  0  ,  0  ,  0  ,  0  ,  1  ]->[ .x. ,  .x. ,.x.];
    [.c.,  1  ,  0  ,  0  ,  0  ,  0  ,  1  ]->[  0  ,  0  , 0 ];
    [.c.,  0  ,  0  ,  0  ,  0  ,  0  ,  0  ]->[  1  ,  0  , 0 ];
    [.c.,  1  ,  0  ,  0  ,  0  ,  0  ,  0  ]->[  0  ,  0  , 0 ];
    [.c.,  0  ,  0  ,  0  ,  0  ,  0  ,  1  ]->[  1  ,  0  , 0 ];
    [.c.,  0  ,  0  ,  0  ,  0  ,  1  ,  0  ]->[  0  ,  0  , 0 ];
    [.c.,  0  ,  1  ,  0  ,  0  ,  0  ,  0  ]->[  1  ,  1  , 0 ];
    [.c.,  0  ,  0  ,  0  ,  0  ,  0  ,  1  ]->[  0  ,  0  , 1 ];
    [.c.,  0  ,  0  ,  0  ,  0  ,  0  ,  0  ]->[  1  ,  0  , 1 ];
    [.c.,  0  ,  1  ,  0  ,  0  ,  0  ,  0  ]->[  0  ,  0  , 1 ];
    [.c.,  0  ,  0  ,  0  ,  0  ,  0  ,  0  ]->[  1  ,  0  , 1 ];
    [.c.,  0  ,  0  ,  0  ,  0  ,  1  ,  0  ]->[  0  ,  0  , 1 ];
    [.c.,  1  ,  0  ,  0  ,  0  ,  0  ,  0  ]->[  1  ,  1  , 1 ];
    [.c.,  0  ,  0  ,  0  ,  0  ,  0  ,  1  ]->[  0  ,  0  , 0 ];
    [.c.,  0  ,  0  ,  1  ,  0  ,  0  ,  0  ]->[  1  ,  1  , 0 ];
    [.c.,  0  ,  0  ,  0  ,  0  ,  0  ,  1  ]->[  0  ,  0  , 1 ];
    [.c.,  0  ,  0  ,  0  ,  1  ,  0  ,  0  ]->[  1  ,  1  , 1 ];
    [.c.,  0  ,  0  ,  0  ,  0  ,  0  ,  1  ]->[  0  ,  0  , 0 ];
    [.c.,  0  ,  0  ,  0  ,  0  ,  0  ,  0  ]->[  1  ,  0  , 0 ];
    [.c.,  0  ,  1  ,  0  ,  0  ,  0  ,  0  ]->[  1  ,  1  , 0 ];
    [.c.,  0  ,  0  ,  0  ,  0  ,  0  ,  1  ]->[  0  ,  0  , 1 ];
    [.c.,  0  ,  0  ,  0  ,  0  ,  0  ,  0  ]->[  1  ,  0  , 1 ];
    [.c.,  1  ,  0  ,  0  ,  0  ,  0  ,  0  ]->[  1  ,  1  , 1 ];
    [.c.,  0  ,  0  ,  0  ,  0  ,  0  ,  1  ]->[  0  ,  0  , 0 ];
    [.c.,  0  ,  0  ,  0  ,  0  ,  0  ,  0  ]->[  1  ,  0  , 0 ];
    [.c.,  0  ,  0  ,  1  ,  0  ,  0  ,  0  ]->[  1  ,  1  , 0 ];
    [.c.,  0  ,  0  ,  0  ,  0  ,  0  ,  1  ]->[  0  ,  0  , 1 ];
    [.c.,  0  ,  0  ,  0  ,  0  ,  0  ,  0  ]->[  1  ,  0  , 1 ];
    [.c.,  0  ,  0  ,  0  ,  1  ,  0  ,  0  ]->[  1  ,  1  , 1 ];
    [.c.,  0  ,  0  ,  0  ,  0  ,  0  ,  1  ]->[  0  ,  0  , 0 ];
END
```

27. No, se requieren 10 entradas y la GAL16V8 sólo tiene ocho.

29. Se saltará el primer estado si hay un vehículo en la calle lateral. El tercer estado siempre se salta.

31.
```
Module      Registro_desplaz_SISO_8_bit

Title       'Registro_desplaz_SISO_8_bit en una GAL16V8'

"Declaración de dispositivo
   Registro      DEVICE        'P16V8'

"Declaración de pines
   Clock,Clear       Pin 1,11;
   Din               Pin 2;
   Q0,Q1,Q2,Q3       Pin 12,13,14,15 ISTYPE 'reg,buffer';
   Q4,Q5,Q6,Qout     Pin 16,17,18,19 ISTYPE 'reg,buffer';

Equations
   [Q0,Q1,Q2,Q3,Q4,Q5,Q6,Qout].CLK = Clock;
   [Q0,Q1,Q2,Q3,Q4,Q5,Q6,Qout].AR  = !Clear;
   Q0   := Din;
   Q1   := Q0.FB;
   Q2   := Q1.FB;
   Q3   := Q2.FB;
```



```
   Q4   := Q3.FB;
   Q5   := Q4.FB;
   Q6   := Q5.FB;
   Qout := Q6.FB;
Test_vectors ([Clock ,Clear, Din]->[Q0,Q1,Q2,Q3,Q4,Q5,Q6,Qout])
              [ .x. ,  0   ,.x.]->[  0 , 0 , 0 , 0 , 0 , 0 , 0,  0  ];
              [ .c. ,  1   , 1 ]->[  1 , 0 , 0 , 0 , 0 , 0 , 0,  0  ];
              [ .c. ,  1   , 0 ]->[  0 , 1 , 0 , 0 , 0 , 0 , 0,  0  ];
              [ .c. ,  1   , 1 ]->[  1 , 0 , 1 , 0 , 0 , 0 , 0,  0  ];
              [ .c. ,  1   , 0 ]->[  0 , 1 , 0 , 1 , 0 , 0 , 0,  0  ];
              [ .c. ,  1   , 1 ]->[  1 , 0 , 1 , 0 , 1 , 0 , 0,  0  ];
              [ .c. ,  1   , 0 ]->[  0 , 1 , 0 , 1 , 0 , 1 , 0,  0  ];
              [ .c. ,  1   , 1 ]->[  1 , 0 , 1 , 0 , 1 , 0 , 1,  0  ];
              [ .c. ,  1   , 0 ]->[  0 , 1 , 0 , 1 , 0 , 1 , 0,  1  ];
END
```

33.
```
Module      Contador_cod_exceso_3 _4_bit

Title       'Contador_cod_exceso_3_4_ bit en una GAL22V10'

"Declaración de dispositivo
   Contador      DEVICE        'P22V10'

"Declaración de pines
   Clock,Reset   Pin 1,2;
   Q0,Q1,Q2,Q3   Pin 12,13,14,15 ISTYPE 'reg,buffer';
   COUNT = [Q3,Q2,Q1,Q0];

Equations
   COUNT.CLK = Clock;

Truth_table  ([Clock,Reset,  COUNT     ]:>[COUNT])
              [ .C. ,  0  ,  .X.       ]:>[^B0011];
              [ .C. ,  1  ,  ^B0000    ]:>[^B0011];
              [ .C. ,  1  ,  ^B0001    ]:>[^B0011];
              [ .C. ,  1  ,  ^B0010    ]:>[^B0011];
              [ .C. ,  1  ,  ^B0011    ]:>[^B0100];
              [ .C. ,  1  ,  ^B0100    ]:>[^B0101];
              [ .C. ,  1  ,  ^B0101    ]:>[^B0110];
              [ .C. ,  1  ,  ^B0110    ]:>[^B0111];
              [ .C. ,  1  ,  ^B0111    ]:>[^B1000];
              [ .C. ,  1  ,  ^B1000    ]:>[^B1001];
              [ .C. ,  1  ,  ^B1001    ]:>[^B1010];
              [ .C. ,  1  ,  ^B1010    ]:>[^B1011];
              [ .C. ,  1  ,  ^B1011    ]:>[^B1100];
              [ .C. ,  1  ,  ^B1100    ]:>[^B0000];
              [ .C. ,  1  ,  ^B1101    ]:>[^B0011];
              [ .C. ,  1  ,  ^B1110    ]:>[^B0011];
              [ .C. ,  1  ,  ^B1111    ]:>[^B0011];

Test_vectors([Clock,Reset]->[COUNT])
              [.C.  ,   0]->[^B0011];
              [.C.  ,   1]->[^B0100];
              [.C.  ,   1]->[^B0101];
              [.C.  ,   1]->[^B0110];
              [.C.  ,   1]->[^B0111];
              [.C.  ,   1]->[^B1000];
              [.C.  ,   1]->[^B1001];
              [.C.  ,   1]->[^B1010];
```

```
                [.C.   ,    1]->[^B1011];
                [.C.   ,    1]->[^B1100];
                [.C.   ,    1]->[^B0011];
END
```

CAPÍTULO 12

1. **(a)** ROM **(b)** RAM

3. El *bus de direcciones* facilita la transferencia de los códigos de dirección a la memoria para acceder a cualquier posición de memoria en cualquier orden para efectuar una operación de lectura o escritura. El *bus de datos* facilita la transferencia de los datos entre el microprocesador y la memoria o la E/S.

5.

	Bit 0	Bit 1	Bit 2	Bit 3
Fila 0	1	0	0	0
Fila 1	0	0	0	0
Fila 2	0	0	1	0
Fila 3	0	0	0	0

7. 512 filas x 128 columnas de 8 bits.

9. Una SRAM almacena bits en flip-flops de forma indefinida mientras que esté aplicada la alimentación. Una DRAM almacena bits en condensadores que deben refrescarse periódicamente para conservar los datos.

11. Véase la Tabla P.10

13. Véase la Figura P.88

TABLA P.10

Entradas		Salidas			
A_1	A_0	O_3	O_2	O_1	O_0
0	0	0	1	0	1
0	1	1	0	0	1
1	0	1	1	1	0
1	1	0	0	1	0

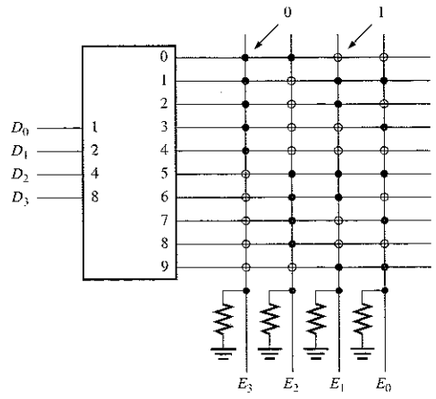


FIGURA P.88

15. Hilos fusibles:1-17, 19-23, 25-31, 34, 37, 38, 40-47, 53, 55, 58, 61, 62, 63, 65, 67, 69.

17. Utilizar ocho memorias RAM de 16k x 4 con dieciséis líneas de dirección. Tres de las líneas de dirección se decodifican para habilitar los chips de memoria seleccionados. Cuatro líneas de datos se usan para cada chip.

19. 8 bits, 64k palabras; 4 bits, 256k palabras.

21. Dirección más baja: $FC0_{16}$
Dirección más alta: FFF_{16}.

23. Un disco duro se formatea en pistas y sectores. Cada pista se divide en una serie de sectores, teniendo cada sector de una pista una dirección física. Normalmente, los discos duros tienen desde unos pocos cientos de pistas hasta unos pocos miles.

25. La cinta magnética tiene un tiempo de acceso mayor que el disco porque debe accederse a los datos de forma secuencial en lugar de forma aleatoria.

27. La suma de comprobación contiene un error.

29. **(a)** ROM 2 **(b)** ROM 1 **(c)** Todas las memorias ROM.

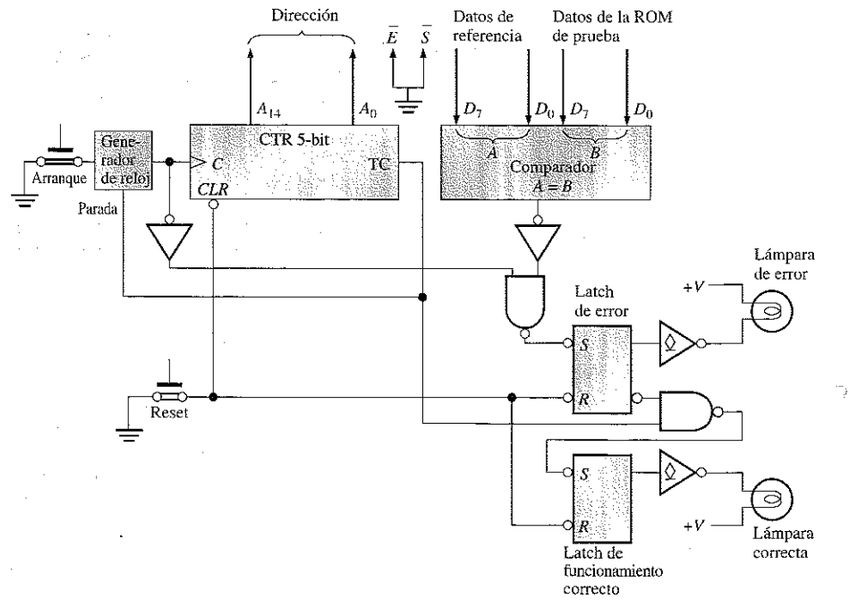
31. 010

33. Una PROM mantiene el código cuando no se alimenta. El código de la PROM no puede cambiarse.

35. Véase la Figura P.89

37. Para acomodar un código de entrada de 5 bits, el registro de desplazamiento C debe cargarse con cinco 0s en lugar de con cuatro. Conectar el pin 14 del 74LS165 a tierra en lugar de a un nivel ALTO.

FIGURA P.89



CAPÍTULO 13

1. 1000, 1010, 1100, 1110, 1110, 1101, 1011, 1001, 0110, 0011, 0010, 0001, 0001, 0010, 0101, 0111, 1011, 1100, 1110, 1110, 1101, 1011, 1001, 0110, 0011, 0001.

3. Véase la Figura P.90

5. 330 kΩ

7. Véase la Figura P.91

FIGURA P.90

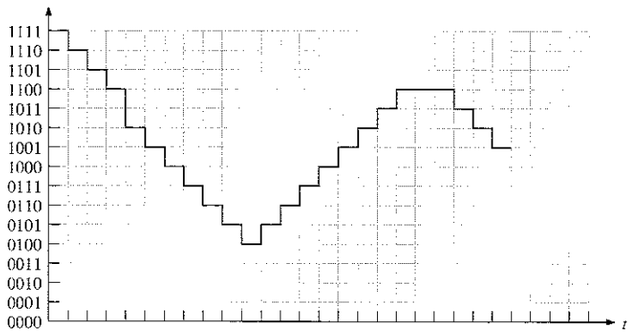
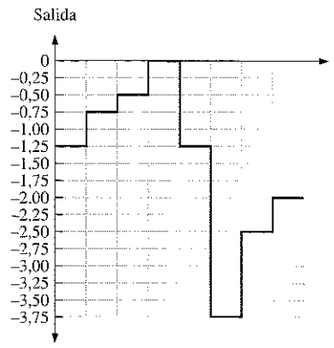


FIGURA P.91



9. **(a)** 14,29% **(b)** 0,098% **(c)** 0,00038%

11. 000, 001, 100, 110, 101, 100, 011, 010, 001, 001, 011, 110, 111, 111, 111, 111, 111, 111, 111, 100

13. 1111, 1100, 1000, 1000, 0011, 0010, 0100, 1110

15. Véase la Tabla P.11

17. Véase la Figura P.92

19. Véase la Figura P.93

21. Véase la Figura P.94

23. El bus local es la colección de buses que establecen directamente la interfaz con el procesador. El bus PCI se usa para dispositivos de expansión y se conecta al bus local a través de un controlador de bus.

25. El bus PCI es un bus de expansión de 32 o 64 bits a 33 o 66 MHz. El bus ISA es un bus de expansión de 8 o 16 bits a 8,33 MHz.

27. DCE hace referencia a un equipo de comunicación de datos, tal como un modem. DTE hace referencia a un equipo terminal de datos, tal como una computadora. Ambos acrónimos se asocian con el estándar RS-232/EIA-232.

29. Siete

31. Un controlador envía datos a dos receptores. Los dos primeros bytes de datos (3F y 41) llegan al receptor con la dirección 001A. Los dos segundos bytes llegan al receptor con la dirección 001B. Las señales para el establecimiento de la comunicación ($\overline{\text{DAV}}$, $\overline{\text{NRFD}}$ y $\overline{\text{NDAC}}$) indican que la transferencia de datos se ha hecho correctamente. Véase la Figura P.95

33. 32,2 kHz

35. 0,71°

TABLA P.11

SAR	Observaciones
1000	Mayor que V_{in}, pone a cero el MSB
0100	Menor que V_{in}, mantiene el 1
0110	Igual que V_{in}, mantiene el estado final

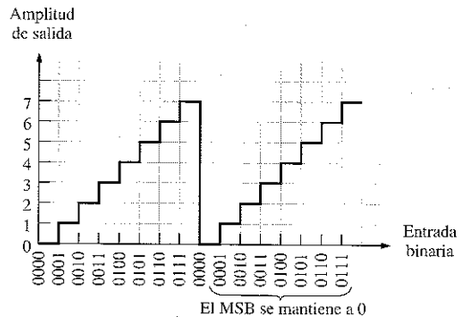


FIGURA P.92

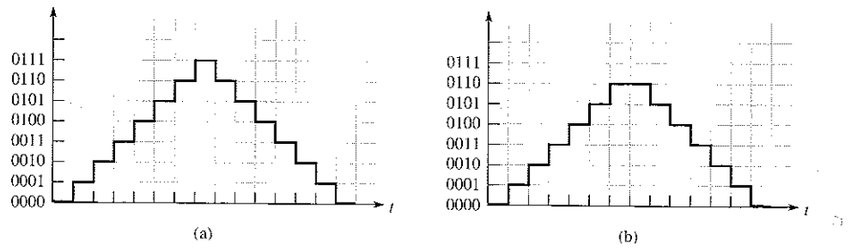


FIGURA P.93

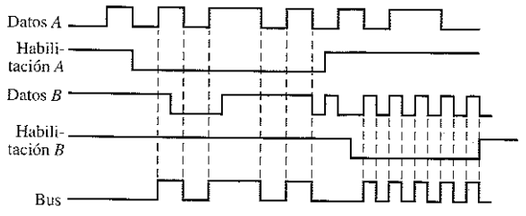
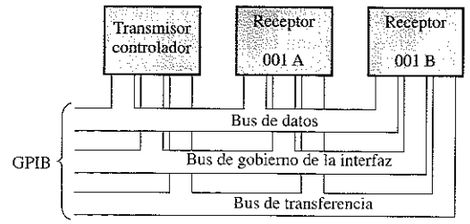


FIGURA P.94

FIGURA P.95



CAPÍTULO 14

1. Unidad aritmético-lógica, matriz de registros, unidad de control.

3. Direcciones: un bus de una dirección que transporta información de direcciones a la memoria o a la E/S.
Datos: un bus de dos direcciones en el que los datos se transfieren al microprocesador o en el que el resultado de una operación se envía desde el microprocesador.
Control: un bus utilizado para coordinar las operaciones del microprocesador y comunicarse con los dispositivos externos.

5. **(a)** 80386 **(b)** aproximadamente 4 Gbytes

7. 68020

9. Carga, decodificación, ejecución.

11. CS, DS, SS, ED, FS, GS

13. AH y AL son registros de 8 bits y representan la parte superior e inferior del registro AX de 16 bits. El EAX es un registro de 32 bits que incluye el registro AX que contiene los 16 bits inferiores.

15. **(a)** Permite que dos instrucciones se ejecuten al mismo tiempo.

(b) El emparejamiento de instrucciones requiere dos unidades de ejecución; el 80486 sólo tiene una.

17. Véase la Figura P.96

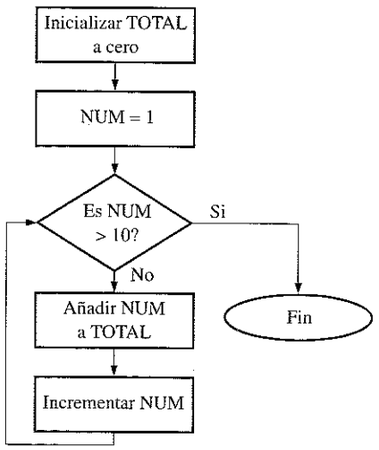


FIGURA P.96

19. La directiva dice al ensamblador que la ubicación apuntada es de 16 bits.

21. *Transferencia de datos:* estas instrucciones se usan para copiar un byte, una palabra (16 bits) o una palabra doble (32 bits) entre varias fuentes y destinos. Un ejemplo es la instrucción MOV.

Aritméticas: estas instrucciones realizan las operaciones de suma, resta, multiplicación y división con números con y sin signo. Un ejemplo es la instrucción ADD.

Manipulación de bits: estas instrucciones incluyen operaciones lógicas, desplazamientos y rotaciones. Un ejemplo es la instrucción OR.

Bucles y saltos: estas instrucciones sirven para alterar la secuencia normal de las instrucciones. Un ejemplo es la instrucción JNZ.

Cadenas: una instrucción para cadenas puede copiar, cargar, almacenar, comparar o explorar una cadena, bien como un byte o una palabra a un tiempo. Un ejemplo es la instrucción MOVSW.

Subrutinas e interrupciones: una subrutina es un miniprograma que se puede utilizar de forma repetida, pero que sólo se programa una vez. Un ejemplo es la instrucción CALL.

*Control del procesador:*estas instrucciones permiten el control directo de algunos de los indicadores del procesado y de algunas otras diversas tareas. Un ejemplo es la instrucción STC.

23. A_0–A_{15}

25. El bit de estado indica al controlador de bus una operación de E/S o de lectura.

27. 400H-4FFH

29. Escritura de memoria.

31. 64k

33. En una E/S por sondeo, la CPU sondea cada dispositivo por turnos, para ver si necesita que se le dé servicio. En un sistema controlado por interrupciones, los periféricos indican a la CPU cuándo necesitan servicio.

35. Una instrucción de programa que invoca una rutina de servicio de interrupciones.

CAPÍTULO 15

1. No, $V_{OH(\text{mín})} < V_{IH(\text{mín})}$

3. 0,15 V en el estado ALTO; 0,25 V en el estado BAJO.

5. Puerta *C*

7. 16 ns

9. Puerta *C*

11. Sí, G_2

13. **(a)** activado **(b)** bloqueado **(c)** bloqueado **(d)** activado

15. Consultar la Figura P.97 para ver un posible circuito.

17. **(a)** ALTO **(b)** flotante **(c)** ALTO **(d)** alta-*Z*

19. **(a)** BAJO **(b)** BAJO **(c)** BAJO

21. Véase la Figura P.98

23. **(a)** $R_p = 198\ \Omega$ **(b)** $R_p = 198\ \Omega$ **(c)** $R_p = 198\ \Omega$

25. ALVC

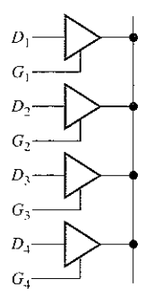
27. **(a)** *A*, *B* to *X*: 9,9 ns
C, *D* to *X*: 6,6 ns

(b) *A* to X_1, X_2, X_3: 14 ns
B to X_1: 7 ns
C to X_2: 7 ns
D to X_3; 7 ns

(c) *A* to *X*: 11,1 ns
B to *X*: 11,1 ns
C to *X*: 7,4 ns
D to *X*: 7,4 ns

29. ECL opera con transistores BJT no saturados.

FIGURA P.97



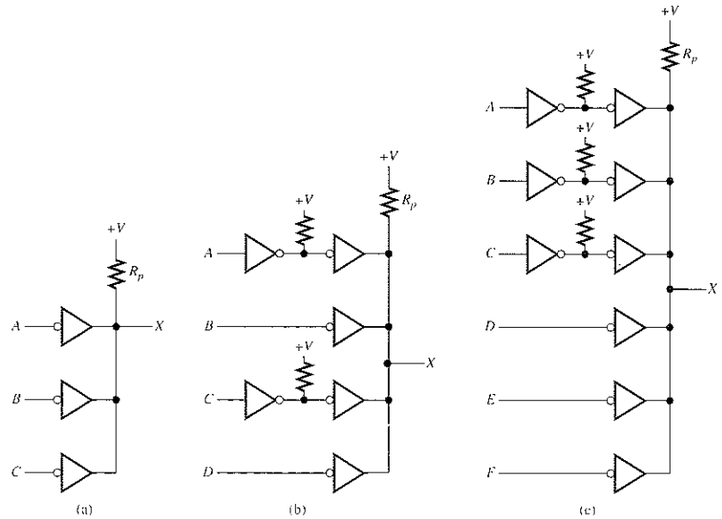


FIGURA P.98

GLOSARIO

ABEL Advanced Boolean Expression Language. Lenguaje de compilación para la programación de PLD; un tipo de lenguaje de descripción hardware.

Acarreo Dígito generado cuando la suma de dos dígitos binarios excede en 1.

Acarreo anticipado Método de suma binaria en que los acarreos de las etapas sumadoras precedentes se anticipan, eliminando de esta manera los retardos en la propagación de los acarreos.

Acarreo serie Método de suma binaria en el que el acarreo de salida de cada sumador se convierte en el acarreo de entrada del sumador siguiente.

Aceptor Dispositivo receptor en un bus.

ACSII American Standard Code for Information Interchange, código estándar americano para el intercambio de información; el código alfanumérico más utilizado.

Acumulador Registro de propósito general utilizado para las operaciones aritméticas y lógicas.

ADC flash o paralelo Convertidor analógico-digital simultáneo.

Adyacencia Característica de las celdas en un mapa de Karnaugh en el que sólo se cambia una variable de una celda a otra inmediata a ella por cualquiera de sus cuatro lados.

Aestable Que no tiene ningún estado estable. Un multivibrador aestable oscila entre dos estados semiestables.

Alfanumérico Que contiene números, letras y otros caracteres.

Álgebra booleana Las matemáticas de los circuitos lógicos.

Almacenamiento Capacidad de los dispositivos digitales para guardar bits. Proceso de conservación de datos para su uso posterior.

Alta-Z Estado de alta impedancia de un circuito triestado, en el que la salida se desconecta del resto del circuito.

ALU (Arithmetic Logic Unit) Unidad aritmético-lógica; el elemento clave de procesamiento de un microprocesador, que realiza operaciones aritméticas y lógicas.

Amplificador operacional Dispositivo con dos entradas diferenciales que tiene una ganancia muy alta, muy alta impedancia de entrada y muy baja impedancia de salida.

Amplitud En un tren de impulsos, la altura o máximo valor del impulso medido desde su nivel más bajo.

Analógico Que es continuo o toma valores continuos, en oposición a tomar valores discretos.

Anchura del impulso Intervalo de tiempo entre los puntos del 50% del valor de los flancos anterior y posterior de un impulso. La duración de un pulso.

AND Operación lógica básica en la que se obtiene una salida verdadera (nivel ALTO) sólo si todas las condiciones de entrada son verdaderas (nivel ALTO).

ANSI American National Standards Institute, Instituto Nacional Americano de Normalización.

Arbitraje del bus El proceso que evita que dos fuentes utilicen un bus al mismo tiempo.

Archivo de documentación La información de una computadora que documenta el diseño final posterior al procesamiento del archivo de entrada.

Archivo de entrada La información introducida en una computadora que describe un diseño lógico, utilizando un lenguaje de programación PLD como el ABEL.

Archivo JEDEC Archivo software estándar generado a partir de un software de compilación, que se emplea en un dispositivo de programación para implementar un diseño lógico de una PLD; también se denomina mapa de fusibles o mapa de celdas.

Arquitectura Disposición funcional interna de los elementos, que dota a un dispositivo de características de operación particulares.

Asignación de direcciones Disposición del espacio de memoria (direcciones) en una computadora.

Asíncrono Que no tiene ninguna relación temporal fija.

Basculación Acción de un flip-flop cuando cambia de estado con cada impulso de reloj.

Base Una de las tres regiones de un transistor bipolar.

BCD Binary Coded Decimal, código decimal binario, código digital en el que cada dígito decimal, de 0 a 9, se representa mediante un grupo de cuatro bits.

BEDO DRAM Memoria de acceso aleatorio dinámica con salida de datos extendida a ráfagas.

Bidireccional Que posee dos direcciones. En un registro de desplazamiento bidireccional, los datos almacenados se pueden desplazar a la derecha o a la izquierda.

Biestable Que tiene dos estados estables. Los flip-flops y los latches son multivibradores biestables.

Binario Que tiene dos valores o estados; describe un sistema de numeración en base dos y utiliza el 1 y el 0 como dígitos.

BIOS Sistema básico de entrada/salida. Conjunto de programas en ROM que establece la interfaz con los dispositivos de E/S en un sistema de computadora.

Bipolar Transistor que tiene dos tipos de portadores de carga opuestos dentro de su estructura.

Bit Dígito binario que puede ser 1 o 0.

Bit de paridad Bit que se añade a cada grupo de bits de información para hacer que el número de unos sea par o impar en dicho grupo de bits.

Bit de signo El bit más a la izquierda de un número binario que designa si el número es positivo (0) o negativo (1).

Bit más significativo (MSB, Most Significant Bit) El bit más a la izquierda de un número entero o código binario.

Bit menos significativo (LSB, Least Significant Bit) Generalmente, el bit de más a la derecha de un número entero o código binario.

BIU Unidad de interfaz de bus. La parte de la CPU que establece la interfaz con los buses del sistema y carga las instrucciones, lee los operandos y escribe los resultados.

BJT Bipolar Junction Transistor, transistor bipolar de unión; dispositivo semiconductor utilizado para conmutación o amplificación. Un BJT tiene dos uniones, la unión base-emisor y la unión base-colector.

Borrado (clear) Entrada asíncrona utilizada para resetear un flip-flop. Cuando se pone un registro o contador en el estado en que contiene solamente ceros.

Buffer Circuito que evita la carga de una entrada o salida.

Buffer triestado Circuito lógico que posee tres estados de salida: ALTO, BAJO y de alta impedancia (abierto).

Bus Conjunto de interconexiones que establece la interfaz entre uno o más dispositivos basándose en una especificación estandarizada.

Bus de control Un conjunto de conductores de una sola dirección que conecta la CPU con otras partes de la computadora, para coordinar sus operaciones y comunicarse con dispositivos externos.

Bus de datos Un conjunto bidireccional de conductores por los que los datos o códigos de instrucciones se transfieren al microprocesador o por los que se envía, desde el microprocesador, el resultado de una operación o cálculo.

Bus de direcciones Generalmente, un grupo de conductores de una sola dirección que va desde el microprocesador a la memoria y que contiene la información de direcciones.

Bus IEEE-488 También denominado GPIB (bus de interfaz de propósito general). Bus paralelo estándar utilizado ampliamente como interfaz de prueba y medida.

Bus IEEE 1349 Bus serie de transferencia de datos a alta velocidad. También conocido como FireWire.

Bus ISA Industry Standard Architecture, arquitectura estándar de la industria. Estándar de bus paralelo interno.

Bus local Bus interno que conecta el microprocesador a la memoria caché, la memoria principal, el co-procesador y el controlador de bus PCI.

Bus PCI Peripheral Control Interconnect, bus de interconexión de control de periféricos. Un estándar de bus paralelo interno.

Byte Grupo de ocho bits.

Cadena Una secuencia contigua de bytes o palabras.

Capacidad (throughput) Velocidad promedio a la que se ejecuta un programa.

Capacidad de palabra El número de palabras que una memoria puede almacenar.

Capacidad Número total de unidades de datos (bits, nibbles, bytes, palabras) que puede almacenar una memoria.

Carácter Símbolo, letra o número.

Carga (load) Introducir datos en un registro de desplazamiento.

Carga unidad Una medida del fan-out. Una entrada de puerta representa una carga unidad para la salida de la puerta, dentro de la misma familia de circuitos integrados.

Cargar (fetch) Proceso de la CPU por el que se obtiene una instrucción de la memoria.

Cartucho Jaz Un dispositivo de almacenamiento. Discos duros encerrados en un cartucho de plástico rígido, con capacidades de almacenamiento de 1 Gbyte o 2 Gbytes.

Cascada Conectar dispositivos "uno tras otro", como cuando se conectan varios contadores de forma que la salida de un contador esté conectada a la entrada de habilitación del siguiente contador.

CCD Charge-Coupled Device, dispositivo de acoplamiento de carga; un tipo de memoria semiconductora que almacena datos en forma de paquetes de carga y a la que se accede en serie.

CD-R CD grabable, dispositivo de almacenamiento de disco óptico en el que los datos se pueden almacenar una vez.

CD-ROM Dispositivo de almacenamiento en disco óptico en el que los datos se almacenan de forma permanente.

CD-RW CD-reescribible. Disco óptico en el que se pueden escribir y sobreescribir muchas veces los datos.

Celda Un punto de cruce de una fila y una columna en un PLD. Una zona de un mapa de Karnaugh que represen-

ta una única combinación de variables en forma de producto. Un elemento de almacenamiento en una memoria.

Celda global Celda programable de una matriz PLD que afecta a todas las OLMC cuando se programan.

Celda local Celda programable de una matriz PLD que afecta a las OLMC individuales cuando se programan.

Ciclo de trabajo La relación entre la anchura de un impulso y el periodo, expresado en porcentaje.

Circuito Disposición de componentes eléctricos y/o electrónicos interconectados de manera que realicen una función específica.

Circuito integrado (CI) Un tipo de circuito en el que todos sus componentes se encuentran integrados en un único chip semiconductor de muy pequeño tamaño.

Circuito secuencial Circuito digital cuyos estados lógicos dependen de una determinada secuencia temporal.

CMOS Complementary Metal-Oxide Semiconductor, metal-óxido semiconductor complementario. Un tipo de circuito de transistores que utiliza transistores MOSFET (metal-oxide semiconductor field-effect, transistorMOS de efecto de campo) de canal-*n* y canal-*p*.

Cociente El resultado de una división.

Codificador Circuito digital que convierte la información a un formato codificado.

Codificador de prioridad Codificador en el que sólo se codifica el dígito de entrada de valor más alto, ignorándose cualquier otra entrada activa.

Código Un conjunto de bits ordenados según un patrón único y utilizados para representar información tal como números, letras y otros símbolos.

Código de operación Código que representa una instrucción particular del microprocesador. Un mnemónico.

Código Gray Código digital sin pesos caracterizado por el cambio de un único bit entre números codificados adyacentes en una secuencia.

Código máquina Instrucciones básicas en código binario que el procesador es capaz de comprender.

Código reubicable Programa que puede desplazarse a cualquier posición dentro del espacio de memoria sin tener que modificar el código básico.

Coherencia Condición de una red lógica en que la diferencia de los tiempos de propagación a través de dos o más caminos de señal de la red pueden producir una salida errónea.

Cola Memoria de alta velocidad que almacena instrucciones o datos.

Cola de espera Archivo o alineación de elementos, por ejemplo, instrucciones precargadas.

Colector abierto Un tipo de salida en los circuitos TTL en el que el colector del transistor de salida se deja desconectado internamente y se encuentra disponible para conectarse a una carga externa que requiera una corriente o tensión relativamente altas.

Colector Una de las tres regiones de un transistor bipolar.

Comparador Circuito digital que compara las magnitudes de dos cantidades y produce una salida que indica la relación entre estas cantidades.

Compilador Software que traduce instrucciones de programas de alto nivel, como BASIC, Pascal o Fortran, a lenguaje máquina.

Complementación Inversión. Un nivel BAJO es el complemento de un nivel ALTO y 0 es el complemento de 1.

Complemento El inverso u opuesto de un número. En el álgebra booleana es la función inversa, que se expresa mediante una barra por encima de la variable. El complemento de 1es 0 y viceversa.

Conexión en cascada Conectar la salida de un dispositivo a la entrada de un dispositivo similar, permitiendo a uno de los dispositivos excitar a otro, para aumentar la capacidad de operación.

Contador Circuito digital capaz de contar sucesos electrónicos, tales como impulsos, avanzando a través de una secuencia de estados binarios.

Contador ascendente/descendente Contador que puede avanzar tanto en una dirección como en otra a lo largo de una secuencia.

Contador asíncrono Un tipo de contador en el que se utiliza como entrada de reloj de cada etapa la salida de la etapa precedente.

Contador de décadas Contador digital de diez estados.

Contador de propagación Un contador asíncrono.

Contador en anillo Registro en el que un determinado patrón de unos y ceros circula continuamente.

Contador Johnson Un tipo de registro en el que se desplaza un patrón específico de unos y ceros a través de las etapas, creando una secuencia patrón única de bits.

Contador síncrono Un tipo de contador en el que todas las etapas utilizan el mismo impulso de reloj.

Contienda de bus Una condición adversa que podría ocurrir si dos o más dispositivos intentan comunicarse a un mismo tiempo a través de un bus.

Contiguo Combinado.

Controlador Instrumento que puede definir a los demás instrumentos conectados al bus como transmisores o receptores durante una transferencia de datos.

Conversión analógico-digital (A/D) Proceso de convertir una señal analógica en digital.

Conversión digital-analógica (D/A) El proceso de convertir una secuencia de códigos digitales a formato analógico.

Convertidor analógico-digital (ADC) Dispositivo utilizado para convertir una señal analógica a formato digital.

Convertidor de código Dispositivo digital que convierte información codificada de una manera en otra codificación distinta.

Convertidor digital-analógico (DAC) Dispositivo utilizado para convertir una entrada digital en una señal analógica.

Coprocesador Microprocesador diseñado con un conjunto de instrucciones limitado que ha sido optimizado para realizar operaciones aritméticas muy rápidamente; generalmente, trabaja junto con un microprocesador de propósito general.

CPU Central processing unit, unidad central de proceso. Uno de los componentes principales de todas las computadoras que controla el funcionamiento interno y procesa los datos.

Diagrama de estados Representación gráfica de una secuencia de estados o valores.

Cronograma o diagrama de tiempos Gráfico de formas de onda digitales que muestra la relación temporal existente entre todas las señales y cómo varía cada una respecto a las restantes.

CUPL Compiler for Universal Programmable Logic, compilador de lógica universal programable; un tipo de lenguaje de descripción hardware.

DAT Cinta digital de audio. Un tipo de formato de cinta magnética.

Datos Información en formato numérico, alfabético o cualquier otro.

DCE Data Communications Equipment, equipo de comunicación de datos.

Debug Programa incluido en el sistema operativo DOS, que permite realizar diversas operaciones sobre archivos. Incluye un programa ensamblador rudimentario.

Década Que se caracteriza por diez estados o valores.

Decimal Describe un sistema de numeración en base diez.

Decodificador Circuito digital que convierte la información codificada en un formato más familiar o no codificado.

Decrementar Disminuir el estado binario de un contador en una unidad.

Demultiplexor (DEMUX) Circuito que conmuta los datos digitales desde una línea de entrada a varias líneas de salida según una secuencia temporal especificada.

Desbordamiento La condición que se produce cuando el número de bits en una suma excede el número de bits de cada uno de los números que se suman.

Desplazar Mover datos binarios de una etapa a otra dentro de un registro de desplazamiento o de otro dispositivo de almacenamiento; mover datos binarios dentro o fuera de un dispositivo.

Detección de errores Proceso de detección de los bits erróneos de un código digital.

Diagrama de estados Una descripción gráfica de una secuencia de estados o valores.

Diferencia El resultado de una sustracción.

Digital Relacionado con los dígitos o con cantidades discretas; que posee un conjunto de valores discretos, en oposición a tener valores continuos.

Dígito Símbolo utilizado para expresar una cantidad.

DIMM Módulo de memoria de terminal doble.

Diodo Dispositivo semiconductor que permite el paso de corriente sólo en una dirección.

Dirección Posición de una determinada celda de almacenamiento o grupo de celdas en memoria. Posición de memoria diferenciada, que contiene un byte.

Dirección base La dirección inicial de un segmento de memoria.

Dirección física Posición real de una unidad de datos en la memoria.

Directiva de ensamblador Instrucción destinada al programa ensamblador. También se denomina pseudooperación.

Disco duro Dispositivo de almacenamiento magnético; normalmente, una pila de dos o más discos rígidos encerrados en un compartimento sellado.

Disco magneto-óptico Dispositivo de almacenamiento que utiliza medios electromagnéticos y un haz láser para leer y escribir datos.

Disco Zip Un tipo de dispositivo de almacenamiento magnético. Disco flexible con una capacidad de 100

Mbytes contenido en un cartucho de plástico rígido de aproximadamente el tamaño de un disquete.

Disipación de potencia El producto de la tensión de alimentación continua y la corriente de alimentación continua en un circuito electrónico. Cantidad de potencia requerida por un circuito.

Disparador Impulso que se utiliza para iniciar un cambio en el estado de un circuito lógico.

Disquete o disco flexible Dispositivo de almacenamiento magnético. Típicamente suele ser un disco flexible de 3,5 pulgadas, con una capacidad de almacenamiento de 1,44 Mbytes, ubicado en el interior de una funda de plástico rígido.

Dividendo En una división, la cantidad que está siendo dividida.

Divisor En una división, la cantidad por la que se divide el dividendo.

DLT Digital Linear Tape (cinta digital lineal). Un tipo de formato de cinta magnética.

DMA Direct Memory Access, acceso directo a memoria. Método para conectar de forma directa un dispositivo periférico con la memoria, sin utilizar la CPU como mecanismo de control.

Dominio Todas las variables de una expresión booleana.

DRAM Dynamic Random-Access Memory, memoria dinámica de acceso aleatorio. Un tipo de memoria semiconductora que utiliza condensadores como elemento de almacenamiento y es una memoria de lectura/escritura volátil.

Drenador Uno de los terminales de un MOSFET.

DTE Data Terminal Equipment, equipo terminal de datos.

Duración del bit Intervalo de tiempo ocupado por un único bit dentro de una secuencia de bits; el periodo del reloj.

DVD-ROM Digital Versatil Disk-ROM (disco digital versátil ROM). También conocido como disco digital de vídeo-ROM. Un tipo de dispositivo de almacenamiento óptico en el que los datos se prealmacenan y que tiene una capacidad mucho mayor que un CD-ROM.

E²CMOS Electrically Erasable CMOS, CMOS eléctricamente borrable. Tecnología de circuitos utilizada para las celdas reprogramables de una GAL.

ECL Emitter-Coupled Logic, lógica de emisor acoplado; un tipo de circuito lógico integrado que se implementa con transistores bipolares no saturados.

EDO DRAM Memoria de acceso aleatorio dinámica con salida de datos extendida.

EEPROM Electrically Erasable Programmable Read-Only Memory, memoria programable de sólo lectura eléctricamente borrable. Un tipo de memoria semiconductora.

EPROM Memoria de sólo lectura programable y borrable. Un tipo de memoria semiconductora.

Ejecutar Proceso de la CPU durante el cual se lleva a cabo una instrucción.

Emisor Una de las tres regiones de un transistor bipolar.

Emparejamiento de instrucciones El proceso de combinar ciertas instrucciones independientes, de modo que puedan ser ejecutadas simultáneamente por dos unidades de ejecución separadas.

Encapsulado DIP (Dual In-line Package) Un tipo de encapsulado de CI cuyos terminales deben insertarse en los taladros de la tarjeta de circuito impreso.

Entrada La señal o línea que entra en un circuito. Señal que controla el funcionamiento de un circuito.

Entrada/Salida (E/S) Un terminal de un dispositivo que puede ser utilizado como entrada o como salida.

Entrada/salida asignada a memoria Un método para direccionar puertos, en el que se asigna a éstos direcciones pertenecientes al espacio de direcciones de memoria de la computadora. La CPU ve los puertos de E/S como posiciones de memoria.

Escritura El proceso de almacenamiento de datos en memoria.

Establecimiento de la comunicación, negociación (handshaking) Método o proceso de intercambio de señales, mediante el que dos dispositivos o sistemas digitales establecen conjuntamente una comunicación.

Etapa Elemento de almacenamiento en un registro.

Exceso 3 Código digital sin peso en el que cada uno de los dígitos decimales se representa mediante un código de 4 bits sumando 3 al dígito decimal y convirtiéndolo a binario.

Expresión booleana Ordenación de variables y operadores lógicos utilizada para expresar el funcionamiento de un circuito lógico.

Fan-out Número de entradas de puertas equivalentes de la misma familia que puede excitar una puerta lógica.

FET Field-Effect Transistor, transistor de efecto de campo.

FIFO First In-First Out, primero en entrar-primero en salir.

Flanco anterior La primera transición de un impulso.

Flanco posterior La segunda transición de un impulso.

Flat pack Encapsulado SMT cuyos terminales se extienden rectos desde el cuerpo.

Flip-flop Circuito básico de almacenamiento que puede almacenar sólo un bit a un tiempo; dispositivo biestable síncrono.

Flip-flop D Un tipo de multivibrador biestable en el que la salida sigue al estado de la entrada D.

Flip-flop disparado por flanco Un tipo de flip-flop en el que los datos se introducen y aparecen en la salida durante el mismo flanco del impulso del reloj.

Flip-flop J-K Un tipo de flip-flop que puede funcionar en los modos de SET, RESET, no cambio y basculación.

Flip-flop maestro-esclavo Un tipo de flip-flop en que los datos de entrada se introducen en el dispositivo durante el flanco anterior del pulso de reloj y aparecen en la salida durante el flanco posterior.

Flip-flop S-R Flip-flop SET-RESET.

FPM DRAM Memoria de acceso aleatorio dinámica en modo página rápido.

Frecuencia El número de impulsos por segundo que hay en una onda periódica. La unidad de frecuencia es el hertzio.

Fuente Dispositivo que envía algo por un bus. Uno de los terminales de un MOSFET.

Fuente de corriente Funcionamiento de un circuito cuando su salida entrega corriente a la carga.

Fusible Elemento programable en ciertos tipos de PLD.

GAL Generic Array Logic, matriz lógica genérica. Dispositivo que tiene una matriz AND reprogramable, una matriz OR fija y macroceldas lógicas programables de salida.

Generación de acarreo El proceso mediante el que se produce un acarreo de salida en un sumador completo cuando las dos entradas son 1.

Glitch Pico de tensión o de corriente de corta duración, no deseado y generalmente producido de forma no intencionada.

GPIB General Purpose Interface Bus, bus de interfaz de propósito general, basado en el estándar IEEE 488.

Habilitar Activar o poner en modo operacional. Una entrada de un circuito lógico que activa su funcionamiento.

Hardware La circuitería y componentes físicos de un sistema de computadora (en oposición a las instrucciones que denominamos software).

HDL Hardware Description Language, lenguaje de descripción hardware; ABEL y CUPL son ejemplos de este tipo de lenguaje.

Hexadecimal Describe un sistema de numeración en base 16.

Histéresis Una característica de los circuitos con umbral de disparo, como el trigger Schmitt, donde el dispositivo se activa y desactiva para diferentes niveles de entrada.

Hoja de especificaciones Documento que especifica los valores de los parámetros y las condiciones de funcionamiento de un circuito integrado o de otro dispositivo.

HPIB Hewlett-Packard Interface Bus, bus de interfaz de Hewlett-Packard; igual que el GPIB (bus de interfaz de propósito general).

I²L Integrated Injection Logic, lógica de inyección integrada; un tipo de tecnología de circuitos integrados.

IEEE Institute of Electrical and Electronic Engineers.

Impulso Cambio brusco desde un nivel a otro, seguido, tras un cierto intervalo de tiempo, por otro cambio brusco que lo devuelve a su nivel original.

Incrementar Aumentar el estado binario de un contador en una unidad.

Indicador Bit que indica el resultado de una operación aritmética o lógica, o que se emplea para alterar una operación.

Indiferente Combinación de literales de entrada que no pueden ocurrir y que se utilizan como unos o ceros en un mapa de Karnaugh.

Inicialización (preset) Entrada asíncrona para inicializar un flip-flop.

Inicio de un nuevo ciclo Sufrir una transición (como ocurre en los contadores) desde el estado final o terminal hasta el estado inicial.

Inmunidad al ruido Capacidad de un circuito para rechazar señales no deseadas.

Instrucción Un paso de un programa de computadora. Unidad de información que le dice a la CPU lo que tiene que hacer.

Interfaz Mecanismo por el que los dos o más dispositivos electrónicos o sistemas se hacen compatibles operacionalmente entre sí, de manera que puedan funcionar adecuadamente juntos.

Interrupción Señal o instrucción que hace que el proceso actual sea temporalmente detenido mientras se ejecuta una rutina de servicio.

Inversión La conversión de un nivel ALTO (1) a un nivel BAJO (0), o viceversa.

Inversor Circuito NOT. Circuito que cambia un nivel ALTO en un nivel BAJO y viceversa.

IP Puntero de instrucción. Un registro especial de la CPU que almacena el desplazamiento correspondiente a la siguiente instrucción que se va a ejecutar.

JEDEC Joint Electronic Device Engineering Council.

Latch Dispositivo digital biestable utilizado para almacenar un bit.

LCCC Leadless Ceramic Chip Carrier, encapsulado SMT que tiene contactos metálicos moldeados en su cuerpo.

LCD Liquid Crystal Display, display de cristal líquido.

Lectura El proceso de recuperar datos de una memoria.

LED Light-Emitting Diode, diodo emisor de luz.

Lenguaje ensamblador Un tipo de lenguaje de programación de computadoras, en el que el código se expresa mediante mnemónicos que representan las instrucciones.

Ley asociativa En la suma (operación OR) y multiplicación (operación AND) de tres o más variables, el orden en que se agrupan las variables no altera el resultado.

Ley conmutativa En la suma (OR) o multiplicación (AND) de dos variables, el orden en el que las variables se suman o multiplican no altera el resultado.

Ley distributiva Ley que dice que si sumamos (operación OR) varias variables y luego multiplicamos (operación AND) el resultado por una única variable, lo que nos queda es equivalente a multiplicar (AND) la variable aislada por cada una de las otras variables y luego sumar (OR) todos los términos.

LIFO Last In-First Out, memoria de tipo primero en entrar-último en salir. Pila de memoria.

Literal Una variable o el complemento de una variable.

Localización de averías Técnica que consiste en identificar, aislar y corregir sistemáticamente cualquier fallo en un circuito o sistema.

Lógica combinacional Combinación de puertas lógicas interconectadas para producir una determinada función booleana sin capacidad de almacenamiento o de memoria.

Lógica En la electrónica digital, la capacidad de los circuitos de puertas de tomar decisiones, en los que un nivel ALTO representa una proposición verdadera y un nivel BAJO representa una proposición falsa.

Lógica positiva Sistema que representa un 1 binario con un nivel ALTO y un 0 binario con un nivel BAJO.

Longitud de palabra El número de bits en una palabra.

LSI Large-Scale Integration, integración a gran escala; un nivel en la complejidad de CI que tienen entre 100 y 9999 puertas equivalente por chip.

Magnitud El tamaño o valor de una cantidad.

Mapa de Karnaugh Disposición de celdas que representa las combinaciones de literales en una expresión booleana y que se utiliza para la simplificación sistemática de la expresión.

Máquina de estados Sistema lógico que exhibe una secuencia de estados condicionada por la lógica interna y las entradas externas. Cualquier circuito secuencial que exhibe una determinada secuencia de estados.

Margen de ruido La diferencia entre la salida máxima a nivel BAJO de una puerta y el máximo nivel de entrada a nivel BAJO aceptable por una puerta equivalente; también, diferencia entre la salida mínima a nivel ALTO de una puerta y el mínimo nivel de entrada a nivel ALTO aceptable por una puerta equivalente. El margen de ruido algunas veces se expresa como un porcentaje de la tensión de alimentación continua.

Matriz En un PLD, una matriz formada por filas de términos productos y columnas de líneas de entrada, con una celda programable en cada unión.

Matriz de memoria Matriz formada por las celdas de memoria colocadas formando filas y columnas.

Matriz de registros Un conjunto de posiciones de almacenamiento temporal dentro del microprocesador, para almacenar datos y direcciones a las que tiene que acceder rápidamente el programa.

Memoria caché Memoria de alta velocidad, relativamente pequeña, que almacena las instrucciones o datos más recientemente utilizados procedentes de la memoria principal, más grande pero más lenta.

Memoria dinámica Memoria semiconductora que tiene celdas de almacenamiento capacitivas que tienden a perder los datos que llevan almacenados más de un cierto periodo de tiempo y, por lo tanto, tienen que ser refrescadas.

Memoria estática Memoria semiconductora volátil que utiliza flip-flops como celdas de almacenamiento y que es capaz de mantener los datos indefinidamente sin necesidad de refresco.

Memoria flash Memoria semiconductora de acceso aleatorio de lectura/escritura no volátil, en la que los datos se almacenan como carga en la puerta flotante de un determinado tipo de FET.

Microcomputadora Computadora en el que la CPU se construye a partir de un único microprocesador.

Microcontrolador Microprocesador especializado diseñado para funciones de control.

Microprocesador Circuito integrado digital que puede ser programado mediante una serie de instrucciones para realizar funciones específicas sobre los datos.

Minimización Proceso por el que se obtiene una expresión en forma de suma de productos o de producto de sumas, conteniendo el menor número posible de términos con el menor número posible de literales por término.

Minuendo Número del que se sustrae otra cantidad.

Mnemónico Instrucción similar al idioma inglés que es convertida por un programa ensamblador en código máquina para que lo use un procesador.

Modem Modulador/demodulador que sirve para conectar dispositivos digitales con sistemas de transmisión analógicos, como las líneas telefónicas.

Modo real Modo de operación de un procesador Intel que emula la memoria de 1 Mbyte del 8086.

Módulo El número de estados en la secuencia de un contador.

Monoestable Que tiene un solo estado estable. Un multivibrador monoestable, o sencillamente monoestable, produce un único impulso en respuesta a una entrada de disparo.

Monotonicidad Característica de un DAC definida por la ausencia de inversiones de paso incorrectas. Un tipo de linealidad digital-analógica.

MOS Metal-Oxide Semiconductor, semiconductor de metal-óxido. Un tipo de tecnología de transistores.

MOSFET Metal-Oxide Semiconductor Field Effect Transistor, transistor de efecto campo de semiconductor metal-óxido.

MSI Medium-Scale Integration, integración a media escala; un nivel en la complejidad de los CI que tienen entre 12 y 99 puertas equivalentes por chip.

Multiplexor (MUX) Circuito que conmuta los datos digitales de distintas líneas de entrada a una única línea de salida según una secuencia temporal especificada.

Multiplicación booleana En el álgebra booleana, la operación AND.

Multiplicador El número que multiplica al multiplicando.

Multiplicando El número que está siendo multiplicado por otro número.

Multitarea Entorno de sistema operativo en el que la computadora parece ejecutar múltiples programas o tareas simultáneamente.

Multivibrador Una clase de circuitos digitales en los que la salida se conecta de vuelta a la entrada (lo cual se denomina realimentación) para producir dos estados estables, un único estado estable o ningún estado estable, dependiendo de la configuración.

Negativa-AND Operación equivalente a una puerta NOR, en la que se obtiene una salida a nivel ALTO cuando todas sus entradas están a nivel BAJO.

Negativa-OR Operación equivalente a una puerta NAND, en la que se obtiene una salida a nivel ALTO cuando una o más entradas están a nivel BAJO.

Nibble Un grupo de cuatro bits.

NMOS *n*-channel Metal-Oxide Semiconductor, semiconductor de metal-óxido de canal *n*.

No volátil Término que describe la memoria que puede mantener los datos almacenados cuando se desconecta la alimentación.

Nodo Punto de conexión común en un circuito, en el que la salida de una puerta se conecta a una o más entradas de puerta.

NOT Operación lógica básica que realiza inversiones.

Notación de dependencia Sistema de notación para los símbolos lógicos que especifica las relaciones de entrada y salida, definiendo una determinada función de forma completa; parte integral de un estándar ANSI/IEEE 91-1984.

Numérico Relativo a los números.

Número en coma flotante Representación de un número basada en la notación científica, en la que el número consta de un exponente y una mantisa.

Octal Describe un sistema de numeración en base ocho.

Offset La distancia en cantidad de bytes de una dirección física desde la dirección base.

OLMC Output-Logic Macrocell, macrocelda lógica de salida. Salida programable de una GAL.

Operando Variable, registro, posición de memoria o valor utilizado como parte de la instrucción en programas escritos en lenguaje ensamblador.

OR Una operación lógica básica en la que una salida verdadera (nivel ALTO) se produce cuando una o más de las condiciones de entrada son verdaderas (nivel ALTO).

OR-exclusiva (XOR) Operación lógica elemental en la que ocurre un nivel ALTO cuando las dos entradas tienen niveles opuestos.

Oscilador Circuito electrónico que está basado en el principio de la realimentación regenerativa y que produce una señal de salida repetitiva. Fuente de señal.

PAL Programmable Array Logic, matriz lógica programable. Dispositivo con una matriz AND programable y una matriz OR fija.

Palabra Unidad completa de datos binarios.

Paralelo En los circuitos digitales, datos que se producen simultáneamente a través de varias líneas. Transferencia o procesamiento de varios bits simultáneamente.

Paridad En relación a los códigos binarios, tener un número par o impar de unos en un grupo de código.

Paridad impar Condición de poseer un número impar de unos en cada grupo de bits.

Paridad par Condición por la que se tiene un número par de unos en cada grupo de bits.

Periférico Dispositivo o instrumento que proporciona servicios de comunicación con alguna computadora, o proporciona servicios o funciones auxiliares a una computadora.

Periódico Describe una señal que se repite a intervalos de tiempo fijos.

Periodo Tiempo requerido por las señales periódicas para repetirse.

Periodo de latencia El tiempo que tarda el sector deseado en colocarse bajo el cabezal, una vez que el cabezal está posicionado sobre la pista deseada de un disco duro magnético.

Peso El valor de un dígito en un número en función de su posición dentro del número.

PIC Controlador de interrupciones programable; maneja las instrucciones de acuerdo con su prioridad.

Pipeline Cuando se aplica a las memorias, una implementación que permite que se inicie una operación de lectura o de escritura antes de que la operación anterior se complete.

PLA Programmable Logic Array, matriz lógica programable. Dispositivo con matrices AND y OR programables.

PLCC Plastic Leaded Chip Carrier, encapsulado STM cuyos terminales se insertan en su cuerpo en forma de J.

PLD Programmable Logic Device, dispositivo lógico programable.

PMOS *p*-channel Metal-Oxide Semiconductor, semiconductor de metal-óxido de canal *p*.

Polarización directa Tensión de polarización que permite a una unión *pn* semiconductora en un transistor o diodo conducir corriente.

Polarización inversa Condición de polaridad del voltaje que evita que una unión *pn* de un transistor o diodo pueda conducir corriente.

PPI Interfaz periférica programable. Circuito integrado especial utilizado para implementar puertos de E/S.

Precarga El proceso de ejecutar instrucciones al mismo tiempo que otras instrucciones son "cargadas", eliminando tiempo de inactividad. Este proceso también se denomina *pipeline*.

Producto de sumas Expresión booleana que consiste simplemente en multiplicar (operación AND) términos suma (operación OR).

Producto El resultado de una multiplicación.

Producto velocidad-potencia Parámetro de funcionamiento que consiste en el producto del retardo de propagación y la disipación de potencia en un circuito digital.

Programa Grupo de instrucciones creado para resolver un problema específico.

Programa ensamblador Programa que convierte los mnemónicos en código máquina.

Programador Instrumento que programa un dispositivo PLD utilizando un archivo JEDEC obtenido de la computadora en el que se esté ejecutando un paquete software HDL (lenguaje de descripción de hardware).

PROM Programmable Read-Only Memory, un tipo de memoria semiconductora programable de sólo lectura. Dispositivo con una matriz AND fija y una matriz OR programable; se utiliza como dispositivo de memoria, pero, generalmente, no como circuito lógico.

Propagación de acarreo Proceso mediante el que se propaga el acarreo de entrada de un sumador completo hasta que se convierte en acarreo de salida, cuando uno o los dos bits de entrada son 1 y el acarreo de entrada es 1.

Pseudo-operación Instrucción dirigida al programa ensamblador (en oposición a las instrucciones dirigidas a un procesador).

Puerta Circuito lógico que realiza una operación lógica específica, tal como AND u OR. Uno de los tres terminales de un transistor MOSFET.

Puerta AND Puerta lógica que produce una salida a nivel ALTO sólo cuando todas las entradas están a nivel ALTO.

Puerta NAND Puerta lógica que produce una salida a nivel BAJO sólo si todas las entradas están a nivel ALTO.

Puerta NOR Puerta lógica en la que la salida es un nivel BAJO cuando al menos una de las entradas está a nivel ALTO.

Puerta NOR-exclusiva (XNOR) Puerta lógica que produce una salida a nivel BAJO sólo cuando las dos entradas tienen niveles opuestos.

Puerta OR Puerta lógica que produce una salida a nivel ALTO cuando una o más entradas están a nivel ALTO.

Puerta OR-exclusiva (XOR) Puerta lógica que produce una salida a nivel ALTO sólo cuando las dos entradas tienen niveles opuestos.

Puerta universal Tanto una puerta NAND como NOR. El término *universal* se refiere a la propiedad de aquellas puertas que permiten que cualquier operación lógica pueda ser implementada mediante ellas o mediante una combinación de puertas de ese tipo.

Puerto Una interfaz física en una computadora, a través de la cual se transfieren datos hacia o desde un periférico.

Puerto de E/S Puerto de entrada/salida. La interfaz entre un bus interno y un periférico.

Puntero El contenido de un registro que almacena una dirección.

QIC Quarter-inch cassette. Un tipo de cinta magnética.

RAM Random-Access Memory, memoria de acceso aleatorio. Memorias semiconductoras volátiles de lectura/escritura.

Realimentación Tensión de salida o porción de ésta que se conecta de vuelta a la entrada de un circuito.

Receptor Instrumento capaz de recibir datos en un bus GPIB (bus de interfaz de propósito general).

Refresco Renovar los contenidos de una memoria dinámica, recargando las celdas de almacenamiento de tipo capacitivo.

Registro Circuito digital capaz de almacenar y desplazar información; típicamente utilizado como dispositivo de almacenamiento temporal.

Registro de desplazamiento universal Registro que tiene capacidad de entradas y salidas serie y paralelo.

Reloj Señal de temporización básica de un sistema digital.

RESET Estado de un flip-flop o latch cuando la salida es 0. La acción de producir un estado de RESET.

Resistencia de pull-up Resistencia que se utiliza para mantener un determinado punto de un circuito a nivel ALTO cuando se encuentra en estado inactivo.

Restador Circuito lógico utilizado para sustraer dos números binarios.

Resto Cantidad sobrante tras una división.

Retardo de propagación Intervalo de tiempo que transcurre entre la transición de entrada y su correspondiente transición de salida en un circuito lógico.

RISC Reduced Instruction Set Computer, cálculo mediante un conjunto de instrucciones reducidas.

ROM Read-Only Memory, memoria semiconductora no volátil de acceso aleatorio.

Salida Señal o línea que sale de un circuito.

Schottky Un tipo de tecnología de circuitos de lógica transistor-transistor.

SCSI Small Computer System Interface (interfaz de sistemas informáticos de pequeño tamaño). Estándar de bus paralelo externo.

SDRAM Memoria de acceso aleatorio dinámica síncrona.

Secuencia Orden en que ocurren varias cosas en una determinada relación temporal.

Secuencia truncada Secuencia que no incluye todos los posibles estados de un contador.

Secuencial Configuración de salida de un PLD en que la salida procede de un flip-flop.

Segmento Un bloque de memoria de 64k.

Seguimiento de señales Técnica de localización de averías mediante la cual se observan las señales paso a paso, comenzando en la entrada y siguiéndolas hasta la salida, o viceversa. En cada punto, las formas de ondas observadas se comparan con la señal correcta que debería haber en ese punto.

Selector de datos Circuito que selecciona los datos de varias entradas a un tiempo según una secuencia y los coloca en la salida; también se denomina multiplexor.

Semisumador Circuito digital que suma dos bits y produce una suma y un acarreo de salida. No puede manejar acarreos de entrada.

Serie Que tiene un elemento seguido de otro, como ocurre en una transmisión serie de bits; que se produce, como los impulsos, en secuencia en vez de simultáneamente.

SET Estado de un flip-flop o latch cuando la salida es 1. La acción de producir un estado SET.

Símbolos duales Negativa-AND es el símbolo dual de la puerta NOR y Negativa-OR es el símbolo dual de la puerta NAND.

SIMM Single-In-line Memory Module, módulo de memoria de una sola fila de terminales.

Síncrono Que tiene una relación temporal fija.

Sintaxis Formato y simbología utilizado en la descripción de la categoría de las funciones.

Síntesis Proceso software por el que se convierte una descripción de circuito a un archivo JEDEC estándar.

SMT Surface-Mount Technology, tecnología de montaje superficial. Técnica de encapsulado de CIs en la que los encapsulados son más pequeños que los de los DIP y se montan en la superficie de una tarjeta de circuito impreso.

Software Programa informático. Programas que dicen a la computadora qué es lo que tiene que hacer para poder realizar un determinado número de tareas.

SOIC Small-Outline Integrated Circuit. Un encapsulado SMT que recuerda un pequeño DIP pero que tiene sus pines con forma de alas de gaviota.

Sondeo El proceso de comprobar una serie de dispositivos periféricos para determinar si alguno de ellos requiere servicio por parte de la CPU.

SRAM Static Random Access Memory, memoria estática de acceso aleatorio; un tipo de memoria volátil semiconductora de lectura/escritura.

SSI Small-Scale Integration, integración a pequeña escala; un nivel en la complejidad de los CT que tienen doce puertas equivalentes por chip.

Subrutina Serie de instrucciones que se pueden agrupar y que un programa puede usar de forma repetida, pero que sólo se escribe una vez.

Suma booleana En el álgebra booleana, la operación OR.

Suma de productos Expresión booleana que consiste simplemente en sumar (operación OR) términos que contienen productos (operación AND).

Suma El resultado que se obtiene cuando se suman dos o más números.

Sumador Circuito lógico que se emplea para sumar dos números binarios.

Sumador completo Circuito digital que suma dos bits y un acarreo de entrada para producir una suma y un acarreo de salida.

Sumando Los dos operandos que se suman en la operación de la adición.

Sumidero de corriente Funcionamiento de un circuito cuando su salida acepta corriente de la carga.

Supresión de cero Proceso de eliminar los ceros anteriores o posteriores en una presentación en display digital.

Sustraendo Número del que se resta el minuendo.

Tabla de verdad Tabla que muestra las entradas y los correspondientes niveles de salida de un circuito lógico.

Tasa de procesamiento La velocidad media con la que se ejecuta un programa.

Temporizador Circuito que puede ser utilizado como monoestable o como oscilador.

Término producto Producto booleano de dos o más literales equivalente a una operación AND.

Término suma Suma booleana de dos o más literales equivalente a la operación OR.

Tiempo de acceso Tiempo que transcurre desde que se aplica un código de dirección válido en las entradas de una memoria hasta que aparecen datos de salida válidos.

Tiempo de búsqueda Tiempo que tarda el cabezal de lectura-escritura de un disco duro en posicionarse sobre la pista deseada para realizar una operación de lectura.

Tiempo de caída Intervalo de tiempo que transcurre entre el punto en que la señal alcanza el 90% de su valor y el punto en que alcanza el 10% durante el flanco negativo de un impulso.

Tiempo de establecimiento Intervalo de tiempo requerido por los niveles de control para estar en las entradas de un circuito digital, como puede ser un flip-flop, antes del flanco de disparo del impulso de reloj.

Tiempo de mantenimiento El intervalo de tiempo requerido por los niveles de control para permanecer en las entradas de un flip-flop después del flanco de disparo del reloj, de manera que se active fiablemente el dispositivo.

Tiempo de subida Tiempo requerido por un impulso positivo para pasar desde el 10% al 90% de su valor total.

Transistor Dispositivo semiconductor que tiene ganancia en tensión y/o corriente. Cuando se utiliza como dispositivo de conmutación se aproxima a un interruptor abierto o cerrado.

Transmisor Instrumento capaz de transmitir datos por un bus GPIB (bus de interfaz de propósito general).

Triestado Un tipo de salida en un circuito lógico que posee tres estados: ALTO, BAJO y alta-Z.

Truncado Acortado.

TTL Transistor-Transistor Logic, lógica transistor-transistor, un tipo de circuito integrado que utiliza transistores de unión bipolares.

ULSI Ultra Large-Scale Integration, integración a ultra gran escala. Un nivel en la complejidad de los CI que tienen más de 100.000 puertas equivalentes por chip.

Unidad de control La parte del microprocesador que proporciona las señales de temporización y control para introducir y extraer datos del microprocesador y para sincronizar la ejecución de instrucciones.

Unidad de ejecución (EU) La parte de la CPU que ejecuta las instrucciones. Contiene la unidad aritmético-lógica (ALU), los registros de propósito general y los indicadores (*flag*).

Unión La frontera entre una región *n* y una región *p* en un BJT.

USB Universal Serial Bus (bus serie universal). Estándar de bus serie externo.

UV EPROM ROM programable borrable por rayos ultravioleta.

Validación Proceso por el que se utiliza un pulso para muestrear la aparición de un suceso, en un determinado instante de tiempo relativo a dicho suceso.

Valor de fin de cuenta Estado final de la secuencia de un contador.

Variable Símbolo utilizado para representar una magnitud lógica que puede tener tanto un valor 1 como 0; generalmente se designa mediante una letra cursiva.

Vectores de prueba Sección de un programa para PLD que consiste en un conjunto de niveles de entrada y de salida resultantes, y que se utilizan para comprobar un determinado diseño lógico antes de convertirlo en hardware.

VLSI Very Large Scale Integration, integración a muy gran escala. Un nivel en la complejidad de los CI que tienen entre 10.000 y 99.999 puertas equivalentes por chip.

Volátil Término que describe una memoria en la que se pierden los datos almacenados cuando se elimina la alimentación.

VXI Interfaz de instrumentación modular.

WORM Write Once-Read Many (una escritura, muchas lecturas). Un tipo de dispositivo de almacenamiento óptico.

Zócalo ZIF Zócalo de fuerza inserción cero. Tipo de zócalo utilizado en la mayoría de los programadores que aceptan un encapsulado PLD.

ÍNDICE ALFABÉTICO

A

ABEL (Advanced Boolean Expression Language), 446, 450, 455, 463, 690, 695, 702
Acarreo, 18, 61, 110
Acarreo anticipado, 338, 397
Acarreo interno, 338
Acarreo serie, 338, 397
Acceso directo a memoria. *Véase* DMA
Aceptor, 397, 855
ACSII, 110
Acumulador, 336, 651, 904, 943
ADC. *Véase* Convertidor analógico-digital
Adyacencia de celdas, 251, 261
Aestable, 520, 532
Álgebra booleana, 126, 186, 204
  expresiones, 208, 221, 252, 261
  ley distributiva, 206, 261
  leyes asociativas, 206, 261
  leyes conmutativas, 206, 261
  multiplicación, 132, 186, 205
  reglas, 208
  simplificación, 218
  suma, 139, 187
Almacenamiento, 22
Almacenamiento de datos paralelo, 506
Almacenamiento magnético, 787
Almacenamiento magneto-óptico, 793
Almacenamiento óptico, 793
ALU, 895, 911, 943
Amplificador operacional, 830
Amplitud, 9, 41
Analizador lógico, 35
Analógico, 4, 41, 826
Anchura del impulso, 9
AND, operación, 16, 41
AND, puerta, 127, 133, 186
AND-cableada, 981
AND-OR, 276, 292
AND-OR-Inversor, 278
ANSI, 186
Arbitraje del bus, 855
Archivo de documentación, 448, 457, 463
Archivo de entrada, 455, 463
Aritmética binaria
  división, 64
  multiplicación, 64
  resta, 63
  suma, 61
Arquitectura, 418, 464, 895, 943
Arquitectura superescalar, 903
ASCII, 98
Asíncrono, 495, 532, 550

B

Basculación, 493, 532, 550
BCD (Binary Coded Decimal), 92, 110
BCD, suma, 93
BEDO DRAM, memoria, 741, 757
BiCMOS, 159
Biestable, 478
Binario, 41
  dígito, 7
  información, 11
  número, 54, 65
  sistema de numeración, 54
BIOS, 900, 944
Bipolar, tecnología, 158
Bit, 41, 54, 736
Bit de paridad, 381, 397
Bit de signo, 68, 110
Bit más significativo. *Véase* MSB (Most Significant Bit)
Bit menos significativo. *Véase* LSB (Least Significant Bit)
BIU (Bus Interface Unit), 906, 907, 944
BJT (transistor de unión bipolar), 973
Bloque de control común, 604
Boole, George, 202
Borrado (clear), 495, 532
Buffer, 423, 464
Buffer de salida triestado, 434, 464
Buffer triestado, 857
Bus, 371, 855
Bus de control, 897, 944
Bus de datos, 478, 738, 897, 944
Bus de direcciones, 738, 896, 944
Bus de interfaz de propósito general. *Véase* GPIB
Bus externo, 863
Bus interno, 860
Bus local, 860
Bus paralelo, 866
Bus serie universal. *Véase* USB
Byte, 71, 110, 736

C

Capacidad de almacenamiento, 631
Capacidad de memoria, 737
Capacidad de palabra, 777
Carga, 963
Carga TTL, 964
Carga unidad, 165, 186
CCD, memoria, 786
CD_R, 794
CD-ROM, 793
CD-RW, 795
Celda, 261, 422, 736
Celeron, procesador, 904
Chip, 27
Ciclo de trabajo, 10
Cinta magnética, 792
Circuito secuencial, 573, 614
Circuitos abiertos, 171, 173, 301, 302
Circuitos integrados, 24

27C16B, memoria EPROM, 767
555, temporizador, 518
68000, microprocesador, 904
68020, microprocesador, 905
68030, microprocesador, 905
68040, microprocesador, 905
68060, microprocesador, 905
74, 159
74121, monoestable no redisparable, 513
74184, convertidor BCD-binario, 367
74185, convertidor binario-BCD, 367
74ABT, 159
74AC, 159
74ACT, 159
74AHC, 159
74AHC74, doble flip-flop D, 497
74AHCT, 159
74ALB, 159
74ALS, 159
74ALVC, 159
74AS, 159
74BCT, 159
74F, 159
74F148, codificador con prioridad, 363
74HC, 159
74HC112, doble flip-flop J-K, 498
74HC147, codificador con prioridad decimal-BCD, 362, 657
74HC151, multiplexor/selector de datos de 8 entradas, 373
74HC154, decodificador 4-líneas a 16-líneas, 380
74HC157A, cuádruple multiplexor/selector de datos de 2 entradas, 372
74HC160, contador de décadas síncrono, 566
74HC161, contador binario de 4 bits, 588
74HC163, contador binario síncrono de 4 bits, 565, 604
74HC164, registro de desplazamiento de 8 bits con entrada serie-salida paralelo, 637, 662
74HC165, registro de desplazamiento de 8 bits con carga paralelo, 640
74HC174A, séxtuple flip-flop, 658
74HC194, registro de desplazamiento bidireccional universal de 4 bits, 646
74HC194, registro de desplazamiento universal, 662
74HC195, registro de desplazamiento de 4 bits de acceso paralelo, 642, 653, 657
74HC58, doble AND-OR, 278
74HCT, 159
74LS, 159
74LS04, inversor séxtuple, 181
74LS08, cuádruple puertas NAND de 2 entradas, 462
74LS122, monoestable redisparable, 515
74LS138, decodificador 3-líneas a 8-líneas, 385
74LS139, doble demultiplexor/decodificador 2-líneas a 4-líneas, 462
74LS190, contador de décadas ascendente/descendente, 571
74LS279, cuádruple latch S-R, 482
74LS283, sumador binario en paralelo de 4 bits, 340
74LS30, puerta NAND de 4 entradas, 181
74LS51, AND-OR-Inversor, 280
74LS75, latch D con entrada de habilitación, 485
74LS83A, sumador binario en paralelo de 4 bits, 340
74LS86, cuádruple puerta XOR, 181
74LS93A, contador binario asíncrono de 4 bits, 557
74LS93A, contador de 4 bits de módulo 16, 558
74LV, 159
74LVC, 159
74LVT, 159
74S, 159
74XX00, cuádruple NAND de dos entradas, 160
74XX02, cuádruple NOR de dos entradas, 160
74XX04, inversor séxtuple, 160
74XX08, cuádruple AND de dos entradas, 160
74XX10, triple NAND de tres entradas, 160
74XX11, triple AND de tres entradas, 160
74XX20, doble NAND de cuatro entradas, 160
74XX21, doble AND de dos entradas, 160
74XX27, triple NOR de tres entradas, 160
74XX30, NAND de ocho entradas, 160
74XX32, cuádruple OR de dos entradas, 160
74XX86, cuádruple OR-exclusiva de dos entradas, 160
74XX86, cuádruple puerta OR-exclusiva, 280
80286, 902
80386, 902
80486, microprocesador, 903
8086/8088, microprocesadores, 906
80X86, microprocesadores, 901
ADC0804, convertidor analógico-digital, 848
características, 956
contadores, 599
contadores en cascada, 599
DAC0808, convertidor digital-analógico, 836
GAL16V8, 441, 686, 688, 708, 712
GAL22V10, 432, 438, 462, 686, 687, 695
Pentium II, 903
Pentium III, 904
Pentium Pro, 903
Pentium, microprocesador, 903, 912
PowerPC, microprocesador, 905
Circuitos integrados, clasificación, 27
Circuitos integrados, encapsulados, 25, 160

Circuitos integrados, fallos internos, 171
Circuitos integrados, tecnologías, 27
CISC, 915
CMOS (Complementary Metal-Oxide Semiconductor), 158, 165, 186
CMOS (Complementary MOS), 27
CMOS, circuitos, 956, 966, 987
CMOS, manipulación, 971
CMOS, niveles lógicos, 956
Cociente, 80
Codificador, 19, 397
8-líneas a 3-líneas, 363
con prioridad, 362, 397
decimal-BCD, 360
Codificador de teclado, 657
Código, 19, 92
Código 8421, 92
Código ausente, 853
Código de operación, 915
Código decimal binario. Véase *BCD*
Código incorrecto, 853
Código máquina, 915, 944
Código reubicable, 908, 944
Códigos alfanuméricos, 97
Cola de ejecución, 906
Cola de instrucciones, 907
Colector abierto, 976
Comparador, 17, 178, 345, 347
Comparador de magnitud, 348
Compilador, 445, 464, 898, 944
Complementación, 124, 186
Complemento, 110, 126
Complemento a 1, 68, 70
Complemento a 2, 68, 70
Comprobador de paridad, 381
Computadora, 445, 894, 898
Condiciones indiferentes, 244, 261
Conexión en cascada, 342, 398, 584
Contador, 23, 509
Contador ascendente/descendente, 569, 614, 699
Contador asíncrono, 550, 552
Contador basado en registro de desplazamiento, 647
Contador binario síncrono, 559, 561
Contador de décadas, 554, 563, 614
Contador de marca temporal, 563
Contador de propagación, 553
Contador en anillo, 649, 653, 657
Contador Johnson, 647
Contador síncrono, 614
Contadores, 602, 699
Contadores en cascada, 584
Contienda de bus, 855, 858
Controlador de bus, 925
Controlador de DMA, 941
Conversión
analógico-digital, 824, 838
BCD a binario, 365
binario a decimal, 57
binario a Gray, 95
binario a hexadecimal, 84
binario a octal, 91
decimal a binario, 58, 60
decimal a hexadecimal, 86
decimal a octal, 89
digital-analógica, 824, 830
Gray a binario, 96
hexadecimal a binario, 84
hexadecimal a decimal, 85
octal a binario, 90
octal a decimal, 89
Conversión analógico-digital, errores, 853
Conversión digital-analógica, errores, 850
Convertidor analógico-digital, 6, 828
características, 836
con pondenación binaria, 831
de doble pendiente, 843
de pendiente simple, 843
de rampa en escalera, 840
de seguimiento, 842
flash (paralelo), 838
por aproximaciones sucesivas, 845
Convertidor de código, 398
BCD-binario, 367
binario-BCD, 367
binario-Gray, 369
Gray.binario, 369
Convertidor de datos
paralelo-serie, 597
serie-paralelo, 654
Convertidor digital-analógico, 6, 828
en escalera R/2R, 833
Coprocesador, 896, 944
Cortocircuitos, 171, 173, 301, 302
CPLD (complex PLD), 444
CPU. *Véase* Unidad central de procesamiento (UCP)
CPU (Central Processing Unit), 894, 899, 923, 944
basada en 8088, 924
memorias, 930
Cronograma. *Véase* Diagrama de tiempos. *Véase* Diagrama de tiempos

D

DAC. *Véase* Convertidor digital-analógico
DAT (Digital AudioTape), 829
DAT, Digital Audio Tape (cinta digital de audio), 792
Datos, 12, 42
paralelo, 13, 42
serie, 13, 42
DCE (Data Communication Equipment), 864
Debug, 917, 944
Década, 555, 614
Decodificación de contadores, 589
Decodificación parcial, 556
Decodificación, glitches, 590
Decodificador, 19, 398
4-líneas a 10-líneas, 355
BCD a 7-segmentos, 357
BCD a decimal, 355
binario, 350
de 4-líneas a 16-líneas, 351
DeMorgan, teoremas de, 211, 294
Demultiplexación, 597
Demultiplexor, 20, 379, 398
Dependencia AND, 605
Dependencia de control, 604
Dependencia de modo, 605
Desbordamiento, 75, 110
Detección de errores, 102, 383
Detector de transiciones de impulsos, 489
Diagrama de estados, 389, 574, 614, 705, 710
Diagrama de tiempos, 11, 125, 130, 550, 562, 566, 584, 638, 641, 644, 646, 649, 654, 655, 868

Digital, 2, 4, 42, 826
DIMM, módulo de memoria, 779
Diodo, 973
DIP (dual in-line package), 25
Dirección de memoria, 737
Direcciones de memoria, 930
Directiva ensamblador, 916
Disco duro, 787, 899
Disco flexible, 790
Disco magneto-óptico, 793
Disipación de potencia, 164, 165, 186, 504, 961
Disparo por flanco, 489
Display de 7-segmentos, 254
Display multiplexor de 7-segmentos, 374
Dispositivo de acoplamiento de carga. *Véase* CCD
Dispositivo lógico programable. *Véase* PLD
Disquete, 790
Dividendo, 80
División, 80
División binaria, 64
División de frecuencia, 506
División sucesiva por 2, 59
Divisor, 80
DLT, Digital Linear Tape (cinta digital lineal), 792
DMA (Direct Memory Access), 941, 944
DMM (Digital Multimeter). *Véase* Multímetro digital
Dominio, 261
DRAM, memoria, 741, 752, 773
tipos, 756
Drenador, 967
Drenador abierto, 970
DSP. *Véase* Procesador digital de señales
DTE (Data Terminal Equipment), 864
DVD-ROM, 795

E

E/S controlada por interrupción, 939
E/S por sondeo, 938
E2CMOS, 421, 464, 991
ECL (Emitter Coupled Logic), 27, 158, 186
ECL, circuitos, 988
Ecuaciones, 451
EDO DRAM, memoria, 741, 757
EEPROM, memoria, 767
EEPROM. memoria, 773
EIA-232, 864
Eliminación del rebote de los contactos, 481
Emparejamiento de instrucciones, 906, 944
Enmascaramiento, 139
Ensamblador, 204
Ensamblador, programa, 897
Entrada/salida multiplexada, 858
EPROM, memoria, 765
EPROM. memoria, 773
Equipo de comunicación de datos. *Véase* DCE
Equipo terminal de datos. *Véase* DTE
EU (Execution Unit), 906, 911
Excitador triestado, 855
Exponente, 72, 110

F

Fan-out, 165, 186, 301, 963
FIFO, memoria, 781
FireWire, bus, 866
Flanco anterior, 9
Flanco posterior, 9
Flip-flop, 22, 478
Flip-flop D disparado por flanco, 491
Flip-flop disparado por flanco, 486
S-R, 486
Flip-flop maestro-esclavo, 499
Flip-flop, características, 502
Flip-flop, tabla de transiciones, 574, 701
Flip-flops, comparación, 505
Formas de onda digitales, 7, 8
FP (flat-pack), 25
FPGA (Field Programmable Gate Array), 444
FPM DRAM, memoria, 741, 756
Frecuencia, 10, 42
Frecuencia máxima de reloj, 504
Fuente, 855, 967
Fuente de alimentación DC, 37
Fuente de corriente, 964, 979
Fusible, 418, 464

G

GAL (Generic Array Logic), 421, 428
GAL (Generic Array Logic), referencia, 431
Generación de acarreo, 339, 398
Generador de funciones, 37, 376
Generador de paridad, 381
Generador de reloj, 925
Glitch, 385, 398, 525, 556, 590
GOTO, instrucción, 707
GPIB (General Purpose Interface Bus), 866
Gray, código, 95, 110, 699

H

Habilitar, 133, 186
Hardware, 915, 945
HDL (Hardware Description Language), 449
Hexadecimal, sistema de numeración, 82
Histéresis, 532
Hoja de características, 341
Hojas de características, 168

I

IEEE, 186
IEEE 1394, 866
IEEE 488, 866
IF-THEN-ELSE, 705
Impulso, 9, 42
Indicador, 912
Indicador de entrada dinámica, 486
Indicador de negación, 124
Inhibir, 133
Inicialización (preset), 495, 532

Inmunidad al ruido, 958
Instrucción, 350
Instrucciones, tipos, 922
Intel Corporation, 900
Interfaz, 824
Interfaz periférica programable. *Véase* PPI
Interrupción, 923, 937, 945
Interrupción software, 940
Inversión, 124, 186
Inversor, 42, 124
ISA, bus, 860, 861
ISP (In-System Programmable), 448

J

Jaz, cartucho, 791
Jaz, unidad, 791
JEDEC, archivo, 446, 457, 464
Johnson, contador, 647

K

Karnaugh, mapa de, 237, 244, 251, 261, 575, 701

L

Latch D con entrada de habilitación, 484
Latch S-R, 478
Latch S-R con entrada de habilitación, 483
LCCC (Leadless ceramic chip carrier), 25
LCD (Liquid Crystal Display), 255
LED (Light Emitting Diode), 144, 254
LED (Light-Emitting Diode), 151
Lenguaje ensamblador, 897, 945
Lenguaje máquina, 897
LIFO, memoria, 783
Linealidad, 836
Literal, 204, 261
Localización de averías, 29, 170, 181, 301, 385, 524, 598, 659, 796, 849
Lógica, 15
Logica combinacional, 687
Lógica combinacional, 274, 282, 425, 430, 577
Lógica monopuerta, 160
Lógica secuencial, 430, 715
Longitud de palabra, 774
LSB (Least Significant Bit), 110
LSI (integración a gran escala), 27

M

Macrocelda lógica de salida. *Véase* OLMC
Manipulación de bits, 922
Mantisa, 72, 110
Mapa de celdas, 446
Mapa de fusibles, 446
Mapa de Karnaugh, 237, 244, 251, 261, 575, 701
Máquina de estados, 573, 614, 699
Margen de ruido, 958
Matrices de puertas programables por campo. *Véase* FPGA
Matriz de memoria, 736, 742, 744, 772
Matriz de registros, 895
Matriz lógica genérica. *Véase* GAL
Matriz lógica programable PAL. *Véase* PAL (Programmable Array Logic)
Matriz lógica programable PLA. *Véase* PLA (Programmable Logic Array)
Matriz programable, 418
Mealy, máquina de estados. *Véase*
Memoria caché, 748
Memoria dinámica, 750
Memoria estática, 742
Memoria flash, 769, 772
Memoria programable de sólo lectura. *Véase* PROM
Memoria semiconductora, 736
Memorias magnéticas, 23
Memorias semiconductoras, 23
Memorias, comparación, 772
Microcontrolador, 895, 945
Microprocesador, 894, 945
buses, 896
Intel, 900, 906
Motorola, 900, 904
programación, 897, 914
Minimización, 239
Mnemónico, 204, 897
Modo combinacional, 432, 689, 690
Modo complejo, 443
Modo página, 753
Modo protegido, 902
Modo ráfaga, operación en, 748
Modo real, 913
Modo secuencial, 434, 689, 690
Modo simple, 441
Módulo, 554
Monoestable, 511, 532
Monoestable no redisparable, 512, 519
Monoestable redisparable, 515
Monotonicidad, 836
Moore, máquina de estados. *Véase*
MOSFET, 966
MOSFET (Metal-Oxide Semiconductor Field-Effect Transistor), 27
MSB (Most Significant Bit), 110
MSI (integración a media escala), 27, 340, 348
Multímetro digital, 37
Multiplexación, 22, 597
Multiplexor, 20, 370, 398
Multiplicación binaria, 64
Multiplicación booleana, 132, 186, 205
Multiplicación sucesiva por 2, 60
Multiplicador, 19, 77
Multiplicando, 77
Multitarea, 902, 945
Multivibrador, 478

N

NAND CMOS, puerta, 969
NAND TTL, puerta, 975
NAND, puerta, 141, 187, 289, 291
Negativa-AND, 294
Negativa-AND, puerta, 149, 186
Negativa-OR, 291
Negativa-OR, puerta, 143, 186
Negociación (handshaking), 855

Nibble, 338, 398, 736
Niveles lógicos, 7, 8, 164, 167
NMOS (N-channel MOS), 27
NMOS, circuitos, 991
Nodo, 301, 314
NOR CMOS, puerta, 969
NOR, puerta, 147, 187, 290, 294
NOR-exclusiva, 281, 346
NOR-exclusiva, puerta, 153, 155
NOT, operación, 16
Notación de dependencia, 373, 398, 604, 662
Numeración en coma flotante, 72
Números binarios, 54
Números con signo, 67, 71
  multiplicación, 77
  resta, 76
  suma, 75
Números decimales, 52
Números hexadecimales, 82
Números octales, 89

O

Octal, sistema de numeración, 89
Offset, 853
OLMC (Output Logic Macrocell), 430, 432, 441, 464, 686, 690
  GAL16V8, 688
  GAL22V10, 687
Operaciones de memoria, 737, 745, 753, 769, 932
Operando, 915
OR, operación, 17
OR, puerta, 135, 187
OR-exclusiva, 345
OR-exclusiva, puerta, 153, 187, 280
Oscilador, 520
Osciloscopio, 29
  analógico, 29
  digital, 33

P

PAL (Programmable Array Logic), 421, 464.
PAL (Programmable Array Logic), referencia estándar, 425
Palabra, 736
Paridad, 102, 110, 381
Paridad impar, 102, 398
Paridad par, 102, 398
PCI, bus, 860, 861
Periférico, 861
Periódico, 10
Periodo, 10, 42
Periodo de bit, 11
Periodo de latencia, 790
Peso, 52, 56, 110
Pila de registros, 783
Pila RAM, 785
Pines, numeración, 26
Pipelinización, 929
PLA (Programmable Logic Array), 419, 464
PLCC (plastic leaded chip carrier), 25
PLD (Programmable Logic Device), 415, 418, 695
PLD (Programmable Logic Device), programación, 445, 447
PLD (Programmable Logic Device), software, 449
PLD complejo. *Véase* CPLD
PMOS, circuitos, 990
PPI (Programmable Peripheral Interface), 936
Precarga, 906
Precisión, 836
Procesador digital de señales, 841
Producto, 77
Producto de sumas, 246, 249, 261
Producto velocidad-potencia, 186, 963.
Programa, 894
Programa ensamblador, 945
Programación interna del sistema. *Véase* ISP
Programador, 446, 464
PROM (Programmable Read-Only Memory), 419, 464
PROM, memoria, 764
Propagación de acarreo, 339, 398
Pseudo-operación, 897, 916
Puerta, 42, 127
Puerta CMOS triestado, 970
Puerta en colector abierto, 976
Puerta en drenador abierto, 970
Puerta TTL con salida triestado, 976
Puerta universal, 289, 314
Puertas lógicas integradas, 157
  tipos, 159
Puerto de E/S, 858
Puerto de E/S asignado a memoria, 936
Puerto de E/S dedicado, 935
Puerto de entrada, 900
Puerto de salida, 900
Puerto paralelo, 863
Puerto serie, 863
Puertos de entrada/salida, 894
Pulsador lógico, 35
Puntero de instrucción, 908

Q

QIC (cartucho de cuarto de pulgada), 792

R

RAM dinámica, memoria. *Véase* DRAM
RAM estática, memoria. *Véase* SRAM, memoria
RAM, memoria, 739, 799, 899
  familia, 741
Realimentación regenerativa, 478
Receptor, 867
Refresco, 741, 754
Registro, 22, 630
Registro base, 651
Registro contador, 651
Registro de aproximaciones sucesivas, 845
Registro de datos, 651
Registro de desplazamiento universal, 646, 662
Registros de desplazamiento, 694
  aplicaciones, 651
  bidireccionales, 644
  con carga paralelo, 640
  de acceso paralelo, 642
  entrada paralelo-salida serie, 639
  entrada serie-salida paralelo, 636
  entrada y salida paralelo, 642
  entrada-salida serie, 631
  funciones, 630
Registros de propósito general, 911, 943
Registros de segmento, 908, 943
Registros indicadores, 943
Reloj, 11, 42
Reloj digital, 593
Reproductor de CD, 828
RESET, estado, 479, 532
Resistencia de pull-up, 364, 398, 970
Resolución, 836
Resta binaria, 63
Resta hexadecimal, 88
Restador, 19
Resto, 59
Retardo de propagación, 502, 553, 962. *Véase* Tiempo de retardo de propagación
Retardo de tiempo, 651
RISC, 915
ROM de máscara, memoria, 758
ROM programable, memoria. *Véase* PROM
ROM, memoria, 739, 759, 796, 899
  familia, 758
  organización, 762
ROM. memoria, 773
RS-232C, 864
RS-422, 865
RS-423, 865
Rutina de servicio, 938

S

Salida triestado, 743
Schottky, 159
SCSI, 870
SDRAM, memoria, 741, 757
Secuencia, 614
Segmento, 908
Seguimiento de señales, 305, 314
Selector de datos, 370, 398
Semisumador, 332, 398
SET, estado, 479, 532
Signo-magnitud, 68, 69
Símbolo de cualificación, 604
Símbolos duales de puerta, 292, 314
Símbolos lógicos, 160, 372
SIMM, módulo de memoria, 779
Síncrono, 559
Sintaxis, 447
Síntesis, 464
SMT (surface mount technology), 42. *Véase* Tecnología de montaje superficial
Software, 445, 465, 915, 946
SOIC (small-outline IC), 25
Sonda lógica, 35
Sondeo, 938
SPP (speed-power product). *Véase* Producto velocidad-potencia
SRAM, memoria, 741, 742, 743, 773, 801
  síncrona de ráfaga, 747
SSI (integración a baja escala), 27
SSOP (shrink small-outline package), 26
Subrutina, 923, 946
Suma binaria, 61
Suma booleana, 139, 187
Suma de comprobación, 796
Suma de pesos, 58, 60
Suma de productos, 222, 239, 249, 261, 438
Suma en BCD, 93
Suma hexadecimal, 87
Sumador, 18, 42
Sumador completo, 333, 398
Sumador en paralelo, 336, 338, 340
Sumadores, expansión, 342
Sumidero de corriente, 964, 979
Supresión de cero, 358, 398

T

Tabla de transiciones de flip-flop, 701
Tabla de verdad, 125, 129, 136, 154, 155, 187, 208, 211, 217, 284, 339, 351, 453, 485, 487, 491, 493, 500, 703
Tabla del estado siguiente, 574, 700
Tasa de procesamiento, 907
Tecnología de montaje superficial, 25
Temporizador 555, 518
Tensión de alimentación continua, 163
Teoremas de DeMorgan, 211, 294
Término producto, 261
Término suma, 261
Tiempo de acceso, 763
Tiempo de asentamiento, 836
Tiempo de bajada, 9
Tiempo de bit, 42
Tiempo de establecimiento, 503, 533
Tiempo de mantenimiento, 503, 533
Tiempo de retardo de propagación, 162, 165, 187
Tiempo de subida, 9
Tierra virtual, 830
Totem-pole, 973, 979, 983
Transferencia de datos, 12
Transferencia de datos, operaciones, 922
Transistor, 973
Transistor de unión bipolar. *Véase* BJT
Transmisión de datos, 383
Transmisión de datos serie, 654
Transmisor, 867
Transmisor receptor asíncrono universal, 655
Trenes de impulsos, 10, 130, 136, 142, 148, 155, 297
Triestado, 855
Trigger Schmitt, 514
TSC (time stamp counter), 563
TSSOP (thin shrink small-outline package), 26
TTL (Transistor-Transistor Logic), 27, 158, 159, 165, 187
TTL Schottky, 978
TTL, circuitos, 956, 973, 978, 987
TTL, entrada no utilizadas, 985
TTL, inversor, 973
TTL, niveles lógicos, 958
TVSOP (thin very small-outline package), 26

U

UAL (Unidad aritmético lógica), 283

UART (Universal Asynchronous Receiver Transmitter). *Véase* Transmisor receptor asíncrono universal
ULSI (integración a ultra escala), 27
Unidad aritmético lógica (UAL), 283
Unidad aritmético-lógica. *Véase* ALU
Unidad central de proceso. *Véase* CPU
Unidad de carga, 965
Unidad de control, 895
Unidad de ejecución. *Véase* EU
Unidad de interfaz de bus. *Véase* BIU
Unidad de memoria, 899
Unidad de procesamiento central (UCP), 347
USB (Universal Serial bus), 866
UV EPROM, memoria, 766

V

Validación, 386, 398
Variable, 204, 261
Vector de prueba, 454
VLSI (integración a muy gran escala), 27

W

WITH, instrucción, 705
WORM, 794

X

XNOR, puerta, 155
XOR, puerta, 153, 178, 187

Z

Zip, disco, 791
Zip, unidad, 791
Zócalo ZIF, 446, 465